(12)

NUSC Technical Report 6659

27 May 1982

AD A117701

IFD: An Implicit Finite-Difference Computer Model for Solving the Parabolic Equation

Ding Lee
George Botseas
Surface Ship Sonar Department

Naval Underwater Systems Center
Newport, Rhode Island / New London, Connecticut

Preface

This report was prepared under NUSC Project No. A65020, *Finite-Difference Solutions to Acoustic Wave Propagation,* Principal Investigator Dr. D. Lee, Code 3342. This work was supported by NAVMAT, Program Manager CAPT D. F. Parrish, Code 08L, Program Element 61152N, Navy Sub/Task ZR00000101, *In-House Laboratory Independent Research.*

The Technical Reviewer for this report was Dr. A. H. Quazi, Code 3331.

Reviewed and Approved: 27 May 1982

W A Von Winkle

W. A. Von Winkle
Associate Technical Director for Technology

The authors of this report are located at the New London Laboratory, Naval Underwater Systems Center, New London, Connecticut 06320.

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER TR 6659	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) AN IMPLICIT FINITE-DIFFERENCE COMPUTER MODEL FOR SOLVING THE PARABOLIC EQUATION		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Ding Lee George Botseas		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Underwater Systems Center New London Laboratory, New London, CT. 06320		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS A65020 61152N ZR00000101
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Material Command, Code 08L Washington, D.C. 20362		12. REPORT DATE 5/27/82 13. NUMBER OF PAGES 100
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Parabolic wave equation	Underwater acoustics
Implicit finite difference	Computer model
Numerical methods	Range independent
Propagation loss	Range dependent

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A general purpose computer model, based on the implementation of an implicit finite-difference scheme, is developed for the solution of the parabolic wave equation. This model can be used to predict acoustic propagation loss in both range-dependent and range-independent environments. An important feature of the model is that it can handle arbitrary surface boundary conditions and an irregular bottom with arbitrary bottom boundary conditions. In the event that the bottom boundary conditions cannot be expressed

20. (Continued)

mathematically, the model has the capability of introducing an artifical absorbing bottom such that, with the appropriate bottom attenuation, the problem becomes solvable. Another important feature of the model is that it can handle horizontal interfaces of layered media. The model is easy to use and easy to modify. Numerical test examples are included to demonstrate the capabilities of the model.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

IFD: AN IMPLICIT FINITE-DIFFERENCE COMPUTER MODEL FOR SOLVING THE PARABOLIC EQUATION

1. INTRODUCTION

The parabolic equation (PE) method for solving a class of range-dependent underwater acoustic wave propagation problems and the split-step Fourier algorithm for numerically solving the parabolic equation were introduced to the acoustics community by Tappert and Hardin.[1] Since then, several research laboratories have implemented the split-step method into various computer programs for modeling sound propagation in the ocean. In ocean environments where sound interacts weakly with the bottom, the split-step method is fast and accurate, but in cases where the bottom interaction is strong, the method is less accurate. It is for this reason that the search for a general purpose and accurate method for solving the parabolic equation continues.

Among the general purpose, accurate numerical methods for solving parabolic wave equations, the authors have found two classes of methods to be promising: finite-differences and numerical ordinary differential equations. In this report, we describe a finite-difference approach to solving the parabolic equation.

If, as the PE solution is marched out in range, the boundary information at the advanced range level can be expressed in terms of the known values at the present range level, then implicit finite-difference (IFD) methods are more desirable than explicit finite-difference methods since IFD methods are faster and unconditionally stable. For these reasons, an IFD method for solving the PE was selected to be programmed into a computer model.

The IFD model presented in this report can be used to predict acoustic propagation loss in both range-dependent and range-independent environments. An important feature of the model is that it can handle arbitrary surface boundary conditions and an irregular bottom with arbitrary bottom boundary conditions. In the event that the bottom boundary conditions cannot be expressed mathematically, the model is capable of introducing artificial bottom attenuation to make the problem solvable. Another important feature of the model is that it can handle horizontal interfaces of layered media.

The formulation of this IFD scheme for the solution of the parabolic wave equation is discussed in the next section; the treatment of the Neumann boundary condition, horizontal interfaces, and attenuation is also included. Section 3 presents the structure of the IFD model; input and output formats are described in detail. Section 4 is devoted to test problems; input run-streams and subroutines are included in the event the reader may be interested in installing the IFD model in his computer.

Program listings of the IFD model can be found in appendix A. Appendix B contains a listing of a program which may be used to plot the output of the

IFD model; this program is annotated and self-explanatory. All programs are written in FORTRAN for the VAX-11/780 computer.

It is requested that the authors be notified if any difficulties with the model are experienced. User contributions that will enhance the model are invited.

2. SOLUTION BACKGROUND

The theoretical derivation of the parabolic equation (PE) and the IFD formula for solving this equation are summarized below. Details concerning the theoretical development of the PE approximation and the stability, consistency, and convergence of the IFD formula can be found in references 2 and 3. Reference 3 also contains a detailed treatment of bottom boundary conditions.

2.1 THEORETICAL DERIVATION

We begin with the Helmholtz equation in the form

$$\nabla^2 p + k_o^2 n^2 p = 0 \quad , \tag{2.1}$$

where

k_o = reference wavenumber = $\frac{\omega}{c_o}$

$n = n(r,z)$ = index of refraction = $\frac{c_o}{c(r,z)}$

p = acoustic pressure

∇^2 = Laplacian operator

c_o = reference sound speed

$c(r,z)$ = sound speed

$\omega = 2\pi f$

f = source frequency.

If we assume a geometry that is cylindrically symmetric, equation (2.1) can be written as

$$\frac{\partial^2 p}{\partial r^2} + \frac{1}{r}\frac{\partial p}{\partial r} + \frac{\partial^2 p}{\partial z^2} + k_o^2 n^2 p = 0 \quad . \tag{2.2}$$

Using the parabolic decomposition technique, $p(r,z) = u(r,z)\, v(r)$, and neglecting the radial dependence field, $v(r)$, we obtain

$$u_{rr} + u_{zz} + 2ik_o u_r + k_o^2(n^2 - 1)\, u = 0 \quad .$$

If we apply farfield and paraxial approximations, the second derivative of u with respect to r may be dropped from the above equation, which results in the parabolic equation for the transmitted field,

$$u_r = \frac{ik_o(n^2 - 1)}{2} u + \frac{i}{2k_o} u_{zz} \quad . \tag{2.3}$$

Equation (2.3) was first introduced to the underwater acoustics community by Tappert.[4] Expressing equation (2.3) in a general form, we have

$$u_r = a(k_o,r,z)u + b(k_o,r,z)u_{zz} \quad , \tag{2.4}$$

where

$$a(k_o,r,z) = \frac{ik_o(n^2 - 1)}{2}$$

$$b(k_o,r,z) = \frac{i}{2k_o}$$

Given an initial field and the appropriate surface and bottom boundary conditions, an initial value problem of equation (2.4) is said to be well posed in the sense of Hadamard[5] if and only if the solution exists, is unique, and depends continuously on the initial values.

2.2. IMPLICIT FINITE-DIFFERENCE FORMULA

Using the Taylor expansion, we obtain a two-level scheme for equation (2.4):

$$\begin{aligned} u(r + k,z) &= \left(1 + k \frac{\partial}{\partial r} + \frac{1}{2!} k^2 \frac{\partial^2}{\partial r^2} + \ldots\right) u(r,z) \\ &= e^{k\frac{\partial}{\partial r}} u(r,z) \quad . \end{aligned} \tag{2.5}$$

If we let $z = mh$, $r = nk$, and $u(r,z) = u(nk,mh) = u_m^n$, where the depth and range increments Δz and Δr are denoted by h and k, respectively, then equation (2.5) can be written as

$$u_m^{n+1} = e^{k\frac{\partial}{\partial r}} u_m^n \quad . \tag{2.6}$$

Expressing equation (2.6) as an IFD formula, we have

$$e^{-\frac{1}{2}k\frac{\partial}{\partial r}} u_m^{n+1} = e^{\frac{1}{2}k\frac{\partial}{\partial r}} u_m^n \quad . \tag{2.7}$$

To solve equation (2.7), we use

$$\left(1 - \frac{1}{2}k\left[a + bD^2\right]\right) u_m^{n+1} = \left(1 + \frac{1}{2}k\left[a + bD^2\right]\right) u_m^n \quad . \tag{2.8}$$

Using a second order difference for the operator D^2 in the z-direction gives

$$-\frac{1}{2}\beta_m^{n+1} u_{m+1}^{n+1} + \left(\alpha_m^{n+1} + \beta_m^{n+1}\right) u_m^{n+1} - \frac{1}{2}\beta_m^{n+1} u_{m-1}^{n+1}$$

$$= \frac{1}{2}\beta_m^n u_{m+1}^n + (\gamma_m^n - \beta_m^n) u_m^n + \frac{1}{2}\beta_m^n u_{m-1}^n \quad , \tag{2.9}$$

where

$$\alpha_m^n = 1 - \frac{1}{2}ka_m^n$$

$$\beta_m^n = b_m^n \frac{k}{h^2}$$

$$\gamma_m^n = 1 + \frac{1}{2}ka_m^n \quad .$$

If we write $X_m^{n+1} = \alpha_m^{n+1} + \beta_m^{n+1}$,

and

$$Y_m^n = \gamma_m^n - \beta_m^n \quad ,$$

equation (2.9) can be expressed in the matrix form

$$
\begin{bmatrix}
X_1^{n+1} & -\frac{1}{2}\beta_1^{n+1} & 0 & 0 & \cdots & 0 & 0 \\
-\frac{1}{2}\beta_2^{n+1} & X_2^{n+1} & -\frac{1}{2}\beta_2^{n+1} & 0 & \cdots & 0 & 0 \\
 & & & \vdots & & & \\
0 & 0 & 0 & 0 & \cdots & X_{m-1}^{n+1} & -\frac{1}{2}\beta_{m-1}^{n+1} \\
0 & 0 & 0 & 0 & \cdots & -\frac{1}{2}\beta_m^{n+1} & X_m^{n+1}
\end{bmatrix}
\bullet
\begin{bmatrix}
u_1^{n+1} \\ u_2^{n+1} \\ \\ u_{m-1}^{n+1} \\ u_m^{n+1}
\end{bmatrix}
=
\begin{bmatrix}
\frac{1}{2}\beta_1^{n+1} u_o^{n+1} \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}\beta_m^{n+1} u_{m+1}^{n+1}
\end{bmatrix}
$$

$$
+
\begin{bmatrix}
Y_1^{n} & \frac{1}{2}\beta_1^{n} & 0 & 0 & \cdots & 0 & 0 \\
\frac{1}{2}\beta_2^{n} & Y_2^{n} & \frac{1}{2}\beta_2^{n} & 0 & \cdots & 0 & 0 \\
 & & & \vdots & & & \\
0 & 0 & 0 & 0 & \cdots & Y_{m-1}^{n} & \frac{1}{2}\beta_{m-1}^{n} \\
0 & 0 & 0 & 0 & \cdots & \frac{1}{2}\beta_m^{n} & Y_m^{n}
\end{bmatrix}
\bullet
\begin{bmatrix}
u_1^{n} \\ u_2^{n} \\ \vdots \\ u_{m-1}^{n} \\ u_m^{n}
\end{bmatrix}
+
\begin{bmatrix}
\frac{1}{2}\beta_1^{n} u_o^{n} \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}\beta_m^{n} u_{m+1}^{n}
\end{bmatrix}
\qquad (2.10)
$$

The two components of the first column vector on the right-hand side (RHS) are the two boundary points at the advanced range level. The two components of the last column vector on the RHS are the two boundary points at the initial range level. This scheme is known as the Crank-Nicolson[6] scheme for variable coefficents.

In a region where the reference sound speed is constant, b_m is also constant. Where b_m is constant, equation (2.10) may be simplified such that the matrix form may be expressed as

$$X\, u^{n+1} = u_*^{n+1} + Y\, u^n + u_*^n \quad ,$$

where

X is a square matrix whose elements are

$$X^{n+1}_{i,i} = \frac{2}{k}\frac{h^2}{b}\left(1 - \frac{1}{2}\,k\,a^{n+1}_i + \beta^{n+1}_m\right)$$

$$X^{n+1}_{i,i+1} = X^{n+1}_{i+1,i} = -1$$

0 elsewhere.

u^{n+1} is a vector whose components are the unknowns u^{n+1}_i at the advanced level.

u^{n+1}_* is a vector having zero components everywhere except for the first and last components, which are the surface and bottom boundary conditions, respectively, at the advanced range level.

Y is a square matrix whose elements are

$$Y^{n}_{i,i} = \frac{2}{k}\frac{h^2}{b}\left(1 + \frac{1}{2}\,k\,a^{n}_i - \beta^{n}_m\right)$$

$$Y^{n}_{i,i+1} = Y^{n}_{i+1,i} = 1$$

0 elsewhere.

u^n is a vector whose components are the known values u^n_i at the present range level.

u^n_* is a vector having zero components everywhere except for the first and the last components, which are the surface and bottom boundary conditions, respectively, at the present range level.

The index i ranges from 1 through m.

2.3 BOUNDARY TREATMENT

An optional homogeneous Neumann boundary condition that has been programmed into the package is described below. However, when other types of boundary conditions arise,[7] the user may program an optional subroutine that will allow the program to accept the existing boundary conditions.

Consider $p_N = 0$, which gives

$$p_z \cos\Theta - p_r \sin\Theta = 0 \quad ,$$

where Θ is the angle of the bottom with respect to the surface. Since u_r satisfies our parabolic wave equation, then the associated boundary condition for equation (2.4) can be described by the second-order ordinary differential equation

$$u_{zz} - \frac{\cot\Theta}{b}\, u_z + \left(\frac{\dfrac{\partial H_o^{(1)}(k_o r)}{\partial r}}{H_o^{(1)}(k_o r)} + a \right) \frac{1}{b}\, u = 0 \quad . \tag{2.11}$$

Expressing equation (2.11) by a second order finite-difference, we have

$$\left(1 - h\,\frac{\cot\Theta}{b}\right) u_{m+1} + \left(h\,\frac{\cot\Theta}{b} - 2 + h^2(ik_o + a)\,\frac{1}{b}\right) u_m + u_{m-1} = 0 \quad .$$

Solving the above equation for u_{m+1} in terms of u_m and u_{m-1}, we find that

$$u_{m+1} = -\frac{P}{Q}\, u_m - \frac{1}{Q}\, u_{m-1} \quad , \tag{2.12}$$

where

$$P = h\,\frac{\cot\Theta}{b} - 2 + h^2(ik_o + a)\,\frac{1}{b}$$

$$Q = 1 - h\,\frac{\cot\Theta}{b} \quad .$$

The coefficients of u_m and u_{m-1} are combined with the appropriate Y and X on both sides of equation (2.10). This mathematical manipulation gives us the advantage of using an unconditionally stable IFD formula.[2]

2.4 INTERFACE TREATMENT

Included in the IFD model is the finite-difference treatment of the horizontal interface, as shown in figure 1. The densities, ρ_1 and ρ_2, are considered to be constant throughout the present treatment. To be consistent with the finite-difference formulation, Δz is denoted by h, and Δr is denoted by k. z_B is the depth of the interface boundary. The subscripts 1 and 2 denote medium 1 and medium 2, respectively. As shown in reference 8, the horizontal interface parabolic equation is

$$u_r = \left(\frac{1}{b_1} + \frac{\rho_1}{\rho_2}\,\frac{1}{b_2}\right)^{-1} \left\{ \left(\frac{a_1}{b_1} + \frac{\rho_1}{\rho_2}\,\frac{a_2}{b_2}\right) u + \frac{2}{h^2}\left[\frac{\rho_1}{\rho_2}\left(u_{m+1}^n - u_m^n\right) - \left(u_m^n - u_{m-1}^n\right)\right] \right\} \quad . \tag{2.13}$$

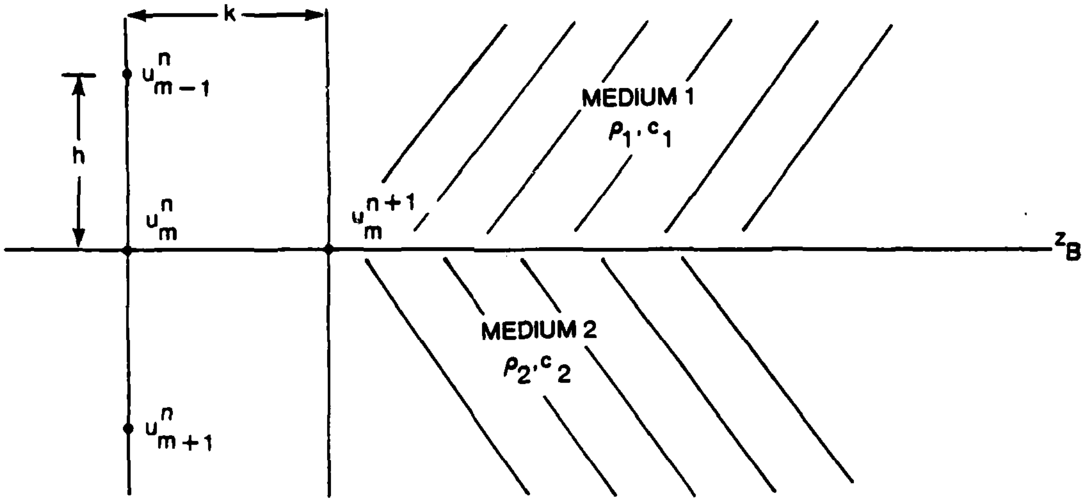


Figure 1. Finite-Difference Treatment of the Horizontal Interface

Using the same implicit finite-difference scheme to solve equation (2.13), we have

$$\left(1 - \frac{1}{2} k \, P_m^{n+1} \, Q_m^{n+1} - k \, P_m^{n+1} \, \tau_{zz}^{n+1}\right) u_m^{n+1}$$

$$= \left(1 + \frac{1}{2} k \, P_m^n \, Q_m^n + k \, P_m^n \, \tau_{zz}^n\right) u_m^n \quad , \tag{2.14}$$

where

$$P = \left(\frac{1}{b_1} + \frac{\rho_1}{\rho_2} \frac{1}{b_2}\right)^{-1} \tag{2.15}$$

$$Q = \left(\frac{a_1}{b_1} + \frac{\rho_1}{\rho_2} \frac{a_2}{b_2}\right) \tag{2.16}$$

$$\tau_{zz}^n \cdot u = \frac{1}{h^2} \left[\frac{\rho_1}{\rho_2} \left(u_{m+1}^n - u_m^n\right) - \left(u_m^n - u_{m-1}^n\right)\right] \quad . \tag{2.17}$$

In the original IFD mathematical formulation,[2] the m-th equation reads

$$\left(-\frac{1}{2}\frac{k}{h^2}\, b_m^{n+1},\ 1-\frac{1}{2}k\, a_m^{n+1}+\frac{k}{h^2}\, b_m^{n+1},\ -\frac{1}{2}\frac{k}{h^2}\, b_m^{n+1}\right)\begin{pmatrix} u_{m-1}^{n+1} \\ u_m^{n+1} \\ u_{m+1}^{n+1}\end{pmatrix}$$

$$=\left(\frac{1}{2}\frac{k}{h^2}\, b_m^{n},\quad 1+\frac{1}{2}k\, a_m^{n}-\frac{k}{h^2}\, b_m^{n},\quad \frac{1}{2}\frac{k}{h^2}\, b_m^{n}\right)\begin{pmatrix} u_{m-1}^{n} \\ u_m^{n} \\ u_{m+1}^{n}\end{pmatrix} \tag{2.18}$$

Writing equation (2.14) in matrix form, we have

$$\left(-\frac{k}{h^2}\, P_m^{n+1},\ 1-\frac{1}{2}k\, P_m^{n+1}\, Q_m^{n+1}+\frac{k}{h^2}\, P_m^{n+1}\left(1+\frac{\rho_1}{\rho_2}\right),\ -\frac{k}{h^2}\, P_m^{n+1}\,\frac{\rho_1}{\rho_2}\right)\begin{pmatrix} u_{m-1}^{n+1} \\ u_m^{n+1} \\ u_{m+1}^{n+1}\end{pmatrix}$$

$$=\left(\frac{k}{h^2}\, P_m^{n},\quad 1+\frac{1}{2}k\, P_m^{n}\, Q_m^{n}-\frac{k}{h^2}\, P_m^{n}\left(1+\frac{\rho_1}{\rho_2}\right),\quad \frac{k}{h^2}\, P_m^{n}\,\frac{\rho_1}{\rho_2}\right)\begin{pmatrix} u_{m-1}^{n} \\ u_m^{n} \\ u_{m+1}^{n}\end{pmatrix} \tag{2.19}$$

Since P is range independent, equation (2.19) may be simplified such that we have

$$\left(-1,\ \frac{h^2}{k}(P)^{-1} - \frac{1}{2}h^2 Q_m^{n+1} + \left(1 + \frac{\rho_1}{\rho_2}\right),\ -\frac{\rho_1}{\rho_2}\right)\begin{bmatrix} u_{m-1}^{n+1} \\ u_m^{n+1} \\ u_{m+1}^{n+1} \end{bmatrix}$$

$$= \left(1,\ \frac{h^2}{k}(P)^{-1} + \frac{1}{2}h^2 Q_m^n - \left(1 + \frac{\rho_1}{\rho_2}\right),\ \frac{\rho_1}{\rho_2}\right)\begin{bmatrix} u_{m-1}^{n} \\ u_m^{n} \\ u_{m+1}^{n} \end{bmatrix} . \qquad (2.20)$$

Incorporating the horizontal interface treatment into the present IFD model requires replacing the row vector elements of both sides of equation (2.18) by the corresponding row vector elements of equation (2.20).

Rewriting equation (2.10), we have

$$
\begin{pmatrix}
X_1^{n+1} & -\frac{\rho_1}{\rho_2} & 0 & 0 & \cdots & 0 & 0 \\
-1 & X_2^{n+1} & -\frac{\rho_2}{\rho_3} & 0 & \cdots & 0 & 0 \\
 & & & \vdots & & & \\
0 & 0 & 0 & 0 & \cdots & X_{m-1}^{n+1} & -\frac{\rho_{m-1}}{\rho_m} \\
0 & 0 & 0 & 0 & \cdots & -1 & X_m^{n+1}
\end{pmatrix}
\begin{pmatrix}
u_1^{n+1} \\ u_2^{n+1} \\ \vdots \\ u_{m-1}^{n+1} \\ u_m^{n+1}
\end{pmatrix}
=
\begin{pmatrix}
u_o^{n+1} \\ 0 \\ \vdots \\ 0 \\ u_{m+1}^{n+1}
\end{pmatrix}
$$

$$
+
\begin{pmatrix}
Y_1^{n} & \frac{\rho_1}{\rho_2} & 0 & 0 & \cdots & 0 & 0 \\
1 & Y_2^{n} & \frac{\rho_2}{\rho_3} & 0 & \cdots & 0 & 0 \\
 & & & \vdots & & & \\
0 & 0 & 0 & 0 & \cdots & Y_{m-1}^{n} & \frac{\rho_{m-1}}{\rho_m} \\
0 & 0 & 0 & 0 & \cdots & 1 & Y_m^{n}
\end{pmatrix}
\begin{pmatrix}
u_1^{n} \\ u_2^{n} \\ \vdots \\ u_{m-1}^{n} \\ u_m^{n}
\end{pmatrix}
+
\begin{pmatrix}
u_o^{n} \\ 0 \\ \vdots \\ 0 \\ u_{m+1}^{n}
\end{pmatrix} ,
\tag{2.21}
$$

where

$$X_m^{n+1} = \frac{h^2}{k}(P)^{-1} - \frac{1}{2}h^2 Q_m^{n+1} + \left(1 + \frac{\rho_m}{\rho_{m+1}}\right)$$

$$Y_m^n = \frac{h^2}{k}(P)^{-1} - \frac{1}{2}h^2 Q_m^n - \left(1 + \frac{\rho_m}{\rho_{m+1}}\right) \quad .$$

The IFD model presented in this report solves equation (2.21).

2.5 ATTENUATION

Attenuation is applied, as suggested by Jensen and Krol,[9] by inserting the units of loss in the imaginary part of the index of refraction as shown below:

$$n^2 = \left(\frac{c_o}{c_i}\right)^2 + i\left(\frac{c_o}{c_i}\right)^2 \frac{\beta}{27.287527} \quad ,$$

where

β is the attenuation in dB/wavelength

c_o is the reference sound speed in m/s

c_i is the speed of sound in m/s at depth i.

3. COMPUTER IMPLEMENTATION

The IFD model that implements the implicit finite difference formula, equation (2.21), has been written in FORTRAN using single precision, complex arithmetic and has been installed on a VAX-11/780 digital computer. A listing of the model can be found in appendix A. The program listing is heavily "commented" in the event the reader wishes to trace through the program logic.

3.1 DESIRABLE FEATURES

The desirable features to be noted in the IFD model are classified below.

A. Generality

The model can solve the parabolic equation in the general form

$$u_r = a(k_0, r,z)\, u + b(k_0,r,z)\, u_{zz} \quad .$$

The following environments can be handled:

- Range-dependent/range-independent
- Shallow water/deep water
- Shallow-to-deep water, deep-to-shallow water, or the combination.

B. Portability

All programs are written in FORTRAN and can be installed in most other computers without much difficulty.

C. Flexibility and Reliability

The user may, at his option, include his own versions of the subroutines that input environmental parameters.

In the event the bottom boundary information is unavailable, our problem, equation (2.4), becomes ill posed. Theoretically, this problem is not uniquely solvable and the computation should not be performed. However, by applying the artificial bottom technique, the bottom may be extended deep enough such that a zero boundary condition results, whereupon the problem becomes artificially well posed and a unique solution exists.

D. Accuracy

The initial local truncation error[2] of the IFD model is

$$E[I] = O(k^3 + kh^2) \quad .$$

Accuracy comes automatically from theory and can be controlled by a PC (predictor-corrector) technique at the expense of computation overhead costs. The present version of the IFD model does not include PC techniques.

E. Special Features

Special features include

- Automatic handling of an irregular bottom
- Acceptance of arbitrary bottom boundary conditions
- Automatic handling of a homogeneous Neumann bottom boundary condition
- Automatic handling of multiple horizontal interfaces and layers
- Capability to introduce an artificial absorbing layer.

3.2 STRUCTURE OF PROGRAM

The IFD computer model consists of a main program and 10 subroutines. The hierarchical structure of the model is as shown in figure 2.

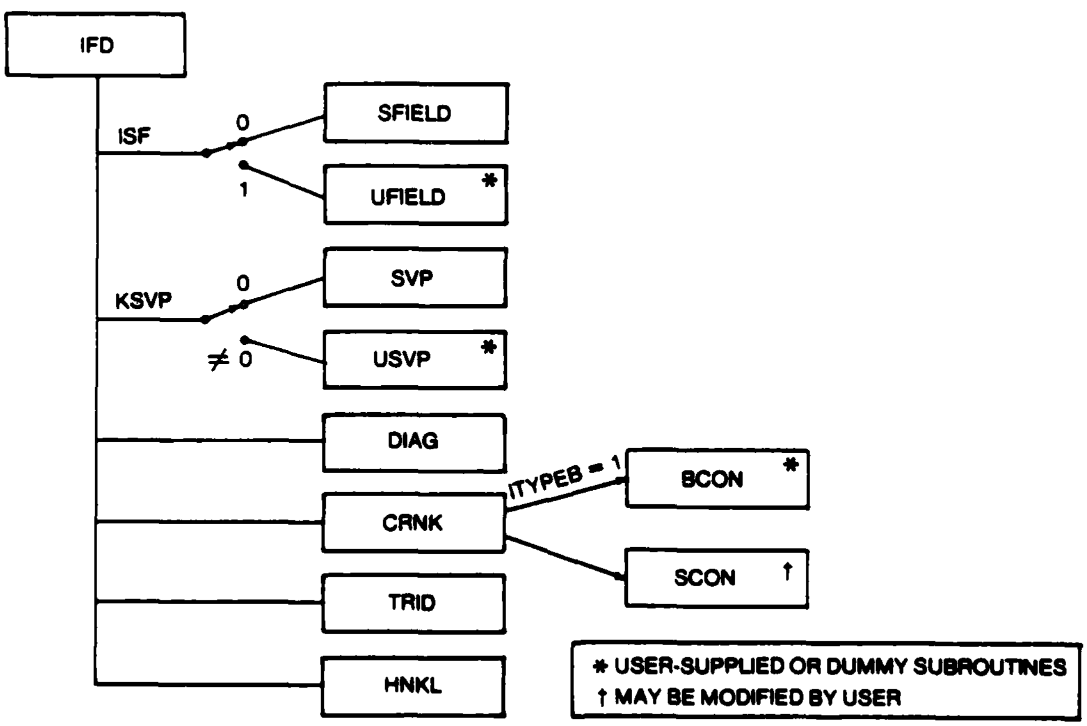


Figure 2. Hierarchical Structure of IFD Model

Those subroutines marked with an asterisk (*) are prepared by the user; the subroutine marked with a dagger (†) may be modified by the user. Input parameters that result in control being transferred to user-supplied subroutines are as shown in the figure. For example, if input parameter ISF = 1, then control is transferred to UFIELD rather than SFIELD.

Linking the program is performed as follows: LINK IFD, CRNK, TRID, DIAG, HNKL, SVP, SFIELD, USVP, UFIELD, BCON, SCON. A brief description of each subroutine follows.

3.2.1 Main Program IFD

IFD is the main program of the model and controls execution of the various subroutines which make up the model. Initially, IFD reads selected input parameters and performs initialization of certain variables. IFD then calls on either subroutine SFIELD or UFIELD to generate the starting field that is to be marched out in range. Selected problem parameters are then printed and, if requested, written in an output file for subsequent use. IFD then calls on subroutine DIAG to compute the main diagonals of the matrices that represent the system of equations at the present and advanced ranges.

After these preliminary procedures have been accomplished, IFD enters a main loop and continues to cycle in the loop until the solution has been marched out to maximum range as requested.

While in the main loop, as the solution is marched out in range, IFD determines whether or not to update the sound speed profile and/or bottom depths. If an update is performed, IFD calls on subroutine DIAG to recompute the main diagonals in the matrices. Whether or not the diagonals have been updated, IFD calls on subroutine CRNK to advance the solution one range step, ΔR, forward. The solution returned by CRNK is then printed and written in an output file as requested by the user. When the solution range has reached the maximum range, the program is terminated. If the solution range has not reached the maximum range, IFD returns to the top of the main loop and repeats the above procedures.

3.2.2 Subroutine SFIELD

If input parameter ISF = 0, main program IFD calls on subroutine SFIELD to generate a Gaussian starting field at zero range. The Gaussian starting field defined by Brock[10] is

$$U(I) = CMPLX\ (PR,\ 0.0)\ ,$$

where

$$PR = GA\left[e^{-\left(\frac{ZM\ -\ ZS}{GW}\right)^2} - e^{-\left(\frac{-ZM\ -\ ZS}{GW}\right)^2}\right]$$

ZM = depth of point in mesh in meters ($I*\Delta z$)

ZS = source depth in meters

GW = Gaussian width $= \dfrac{2}{FK}$

FK = reference wavenumber

GA = Gaussian amplitude $= \dfrac{1}{GW} \left(\dfrac{2}{FK}\right)^{\frac{1}{2}}$.

3.2.3 Subroutine UFIELD

If input parameter ISF = 1, main program IFD calls on user-written subroutine UFIELD to generate the starting field. UFIELD must load N values of the complex starting field in array U.

If ISF = 0, UFIELD is not called and may be a dummy subroutine.

If the user-supplied starting field is an elliptic solution, then the starting field must be divided by the Hankel function and the solution field multiplied by the Hankel function before computing transmission loss. The above procedure can be accomplished automatically by setting input parameter IHNK = 1.

3.2.4 Subroutine SVP

When the range of the solution is equal to the range of the next sound velocity (speed) profile (input parameter RSVP), a new sound speed profile is inputted. If input parameter KSVP = 0, then subroutine SVP is called upon to read the next sound speed profile from the input runstream. SVP then loads the six data tables shown in figure 3.

Table ZLYR contains the maximum depth of each layer as measured from the surface. Tables RHO and BETA contain the density and attenuation, respectively, in each layer. Variable NLYR is the number of layers inputted. The sound speed profile for each layer is stored in tables ZSVP and CSVP. NSVP is the total number of points stored in these tables. If an error is detected while inputting the profiles, NSVP is set to 0. Table IXSVP contains indexes that point to the last sound speed profile value in each layer.

At the present stage of development, linear interpolation of sound speed values is performed only in depth. Changes in the profiles in range are abrupt, with no interpolation performed. Interpolation of the sound speed values is performed in subroutine DIAG.

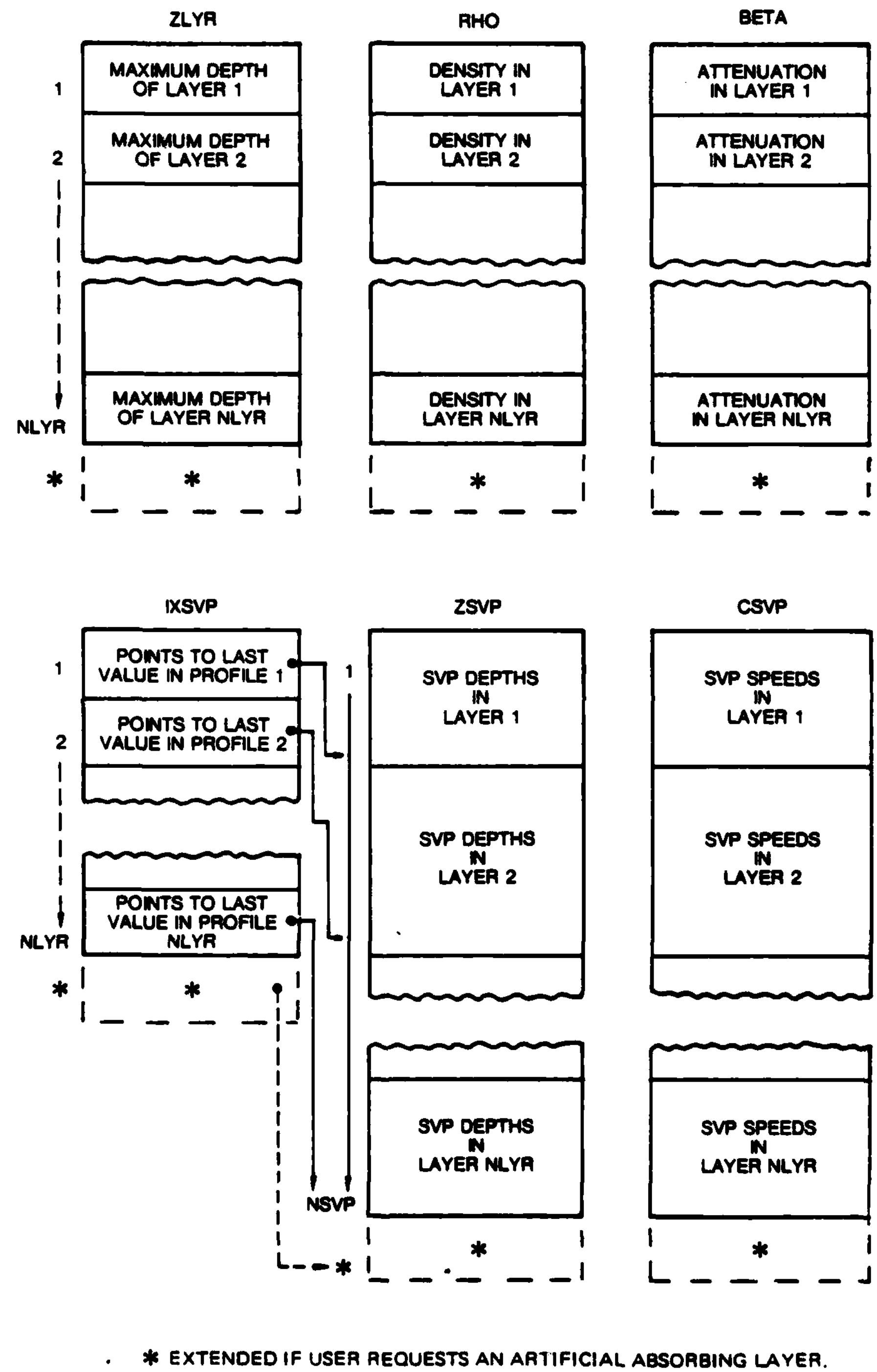


Figure 3. SVP Tables

3.2.5 Subroutine USVP

If input parameter KSVP ≠ 0, then subroutine USVP is called to supply an updated SVP at each step in range. Subroutine USVP must be prepared by the user, who must load the six tables described in the previous section. Variables NLYR and NSVP must also be loaded by the user.

Variable KSVP may be used in a computed GOTO statement to transfer control within user program USVP. When the user no longer needs USVP, KSVP must be set to zero within USVP. The last profile entered will be used until the solution range is equal to the next RSVP. If KSVP is not set to zero, then USVP will be called until the solution range is equal to RSVP, the range of the next profile. With this option, the user can generate a new profile at each range step. Sound speed profile values interpolated in range may be entered by the user in this manner. When the next solution range is equal to the next RSVP, either SVP or USVP, depending on the next KSVP, is called to input the next profile.

3.2.6 Subroutine DIAG

Subroutine DIAG computes the range-dependent and depth-dependent main diagonals of the matrices that represent the system of equations at the present and advanced ranges, as shown in equation (2.21).

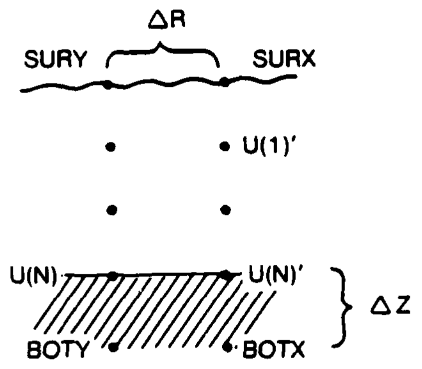
Prior to computing the diagonals, DIAG determines the values of sound speed, density, and attenuation to be used at each depth represented by the corresponding row of each matrix. Linear interpolation of sound speed in depth is performed as required.

3.2.7 Subroutine CRNK

Subroutine CRNK computes the right-hand side of the system of equations, D(i); determines bottom type; sets up bottom conditions at the present and advanced ranges; and then calls on subroutine TRID to solve the tridiagonal system of equations. If the user is supplying surface conditions, then CRNK calls on SCON to provide SURY and SURX, which are added to the computation of D(1). SURY and SURX are the values of the field at the surface at the present and advanced ranges, respectively. If the user is supplying bottom conditions, then CRNK calls on user-written subroutine BCON for these conditions. A summary of the treatment of the bottom follows in figure 4.

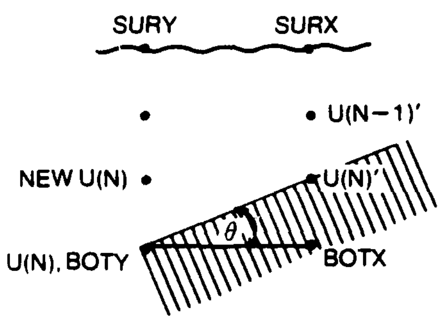
Variables U(1) through U(N) represent the known values of the field at the present range. Variables U(1)′ through U(N)′ represent the values of the field at the advanced range and are to be determined. D(N) is the right-hand side of row N in the system of equations. D(1) through D(N-1) are computed prior to performing the operations summarized on the following pages. BOTY and BOTX are the values of the field at the bottom at the present and advanced ranges, respectively. ITYPEB is an input parameter for selecting the type of bottom treatment to be performed.

ITYPEB = 0, RIGID BOTTOM, FLAT, $\theta = 0.0°$



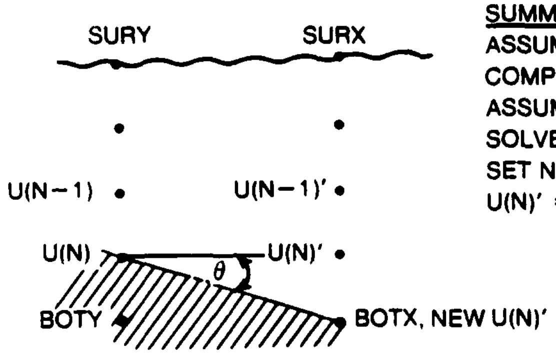
SUMMARY
SET BOTY = U(N)
COMPUTE D(N)
ASSUME BOTX = U(N)'
SOLVE FOR U(1)' THRU U(N)'

ITYPEB = 0, RIGID BOTTOM, SLOPES UP, $\theta < 0.0°$



SUMMARY
SET BOTY = U(N)
SET N = N−1
COMPUTE D(N)
ASSUME BOTX = f(U(N)', U(N−1)', V_1, V_2, ...,V_K)
SOLVE FOR U(1)' THRU U(N)'

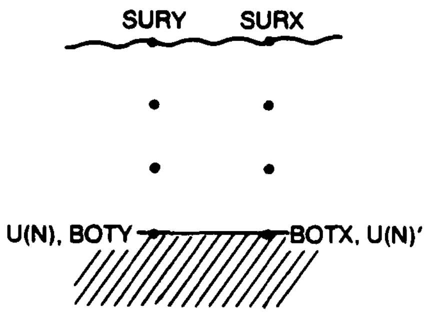
ITYPEB = 0, RIGID BOTTOM, SLOPES DOWN, $\theta > 0.0°$



SUMMARY
ASSUME BOTY = f(U(N), U(N−1), V_1, V_2, ... ,V_K)
COMPUTE D(N)
ASSUME BOTX = f(U(N)', U(N−1)', V_1, V_2, ...,V_K)
SOLVE FOR U(1)' THRU U(N)'
SET N = N+1
U(N)' = f(U(N−1)', U(N−2)', V_1, V_2, ...,V_K)

Figure 4a. Summary of Bottom Treatment

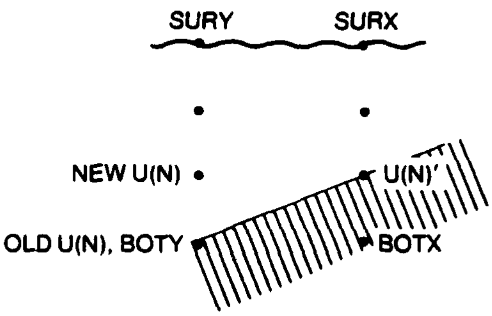
ITYPEB = 1, USER BOTTOM, FLAT, $\theta = 0.0°$



SUMMARY

BCON SETS BOTY = U(N)
BCON SUPPLIES BOTX
N = N − 1
COMPUTE D(N)
SOLVE FOR U(1)' THRU U(N)'
N = N + 1
SET U(N)' = BOTX

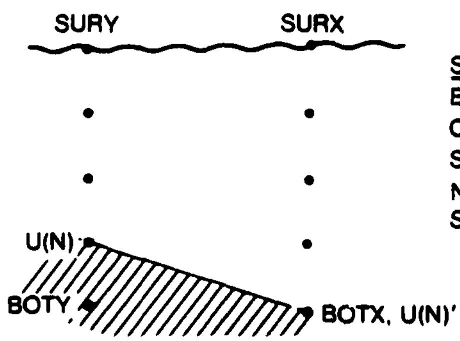
ITYPEB = 1, USER BOTTOM, SLOPES UP, $\theta < 0.0°$



SUMMARY

BCON SETS BOTY = U(N)
BCON SUPPLIES BOTX
N = N − 1 (DELETE A POINT)
COMPUTE D(N)
SOLVE FOR U(1)' THRU U(N)'

ITYPEB = 1, USER BOTTOM, SLOPES DOWN, $\theta > 0.0°$



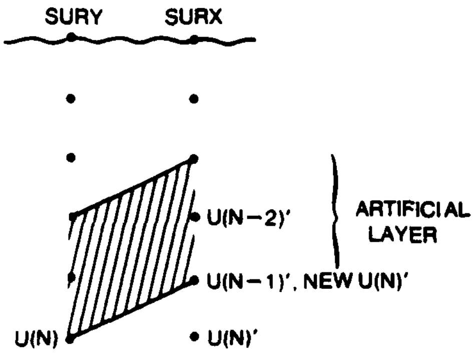
SUMMARY

BCON SUPPLIES BOTY AND BOTX
COMPUTE D(N)
SOLVE FOR U(1)' THRU U(N)'
N = N + 1 (ADD A POINT)
SET U(N)' = BOTX

Figure 4b. Summary of Bottom Treatment

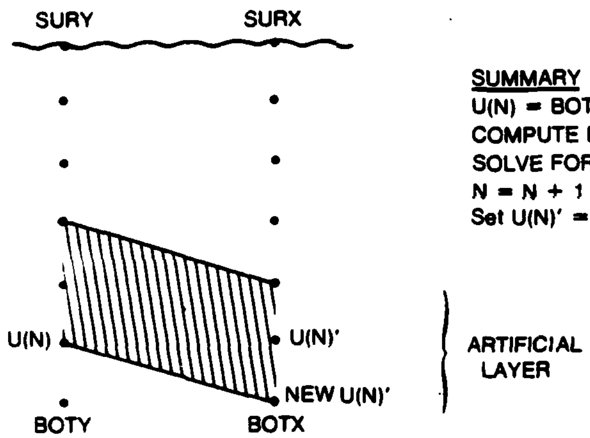
ITYPEB = 2, ARTIFICIAL ABSORBING LAYER, FLAT θ = 0.0°
SEE ITYPEB = 3

ITYPEB = 2, ARTIFICIAL ABSORBING LAYER, SLOPES UP, θ<0.0°



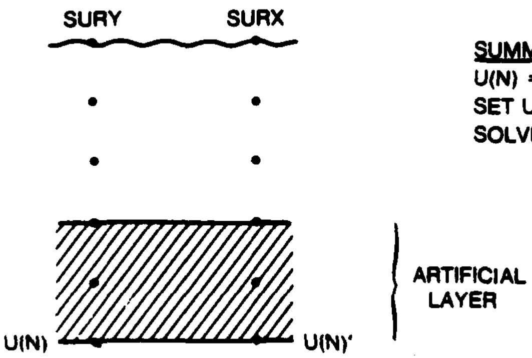
<u>SUMMARY</u>
U(N) = 0.0
Set U(N−1)' = 0.0
Set U(N)' = 0.0
SOLVE FOR U(1)' THRU U(N−2)'
N = N − 1 (DELETE A POINT)

ITYPEB = 2, ARTIFICIAL ABSORBING LAYER, SLOPES DOWN, θ>0.0°



<u>SUMMARY</u>
U(N) = BOTY = BOTX = 0.0
COMPUTE D(N)
SOLVE FOR U(1)' THRU U(N)'
N = N + 1 (ADD A POINT)
Set U(N)' = 0.0

ITYPEB = 3, ARTIFICIAL ABSORBING LAYER, FLAT, θ = 0.0°



<u>SUMMARY</u>
U(N) = 0.0
SET U(N)' = 0.0
SOLVE FOR U(1)' THRU U(N−1)'

Figure 4c. Summary of Bottom Treatment

3.2.8 Subroutine BCON

If input parameter ITYPEB = 1, then subroutine CRNK calls on user-written subroutine BCON to supply BOTY and BOTX, the values of the field on the bottom at the present and advanced ranges, respectively. The treatment of the user bottom was described earlier in the description of subroutine CRNK.

3.2.9 Subroutine SCON

If the user wishes to supply values for SURY and SURX, the values of the field at the surface at the present and advanced ranges, respectively, he may do so by rewriting subroutine SCON. At present, subroutine SCON sets SURY and SURX to 0.0.

3.2.10 Subroutine TRID

Subroutine TRID, a specialized version of subroutine TRIDAG presented in reference 11, solves a system of N linear simultaneous equations having a tridiagonal coefficient matrix. As shown in equation (2.21), the lower diagonal coefficient at the advanced range in rows 2 through N is -1. The upper diagonal coefficient in rows 1 through N-1 is the negative of the ratio of the densities of the media above and below the receiver depth represented by each row. If a receiver lies entirely within the same medium, then the upper diagonal coefficient for that row is also -1. The main diagonal is computed in routine DIAG.

3.2.11 Complex Function HNKL

HNKL computes the Hankel function. The algorithm for computing the Hankel function is described in reference 12.

3.3 USER'S GUIDE

The next two subsections describe input and output formats in detail. Test examples showing sample input runstreams and user-written subroutines can be found later in this report.

Since plotting facilities vary from one activity to another, it was decided that the plotting program should not be included in the IFD model. A separate plot program has been included in appendix B for reference purposes.

3.3.1 Input Format

Prior to executing the IFD model, input card images containing problem parameters must be stored in file IFD.IN. File IFD.IN is assigned to FORTRAN unit number NIU in the main program. If the user prefers to input problem parameters on cards, then parameter NIU should be equated to the card reader unit

number, and the statement which assigns file IFD.IN should be removed from the main program. In either case, the input runstream is prepared in free format as follows:

CARD	CONTENTS
1	FRQ, ZS, CO, ISF, RA, ZA, N, IHNK, ITYPEB, ITYPES

where

FRQ = frequency (Hz)

ZS = source depth (m)

CO = reference sound speed (m/s)

If CO = 0.0, CO is set to the average sound speed in the first layer.

ISF = starting field flag.

0 = Gaussian starting field is generated.

1 = user prepares starting field. See subroutine UFIELD.

If ISF = 0, RA is set to zero.

RA = horizontal range from source to starting field (m).

If ISF = 0, RA is set to 0.0.

ZA = depth of starting field at range RA (m). If ZA = 0.0, ZA is set to the maximum depth of the bottom-most layer in the first profile.

If ITYPEB = 2 or 3 and ZA = 0.0, ZA is set to (4/3)* maximum depth of the bottom-most layer. If ITYPEB = 2 or 3 and ZA ≠ 0.0, the artificial bottom layer is extended to ZA meters provided that ZA is greater than or equal to the maximum depth of the bottom-most layer in the first profile.

N = Number of equispaced receivers in the starting field. If N = 0, N is set so that the receiver depth increment is less than or equal to 1/4 wavelength. If N is greater than MXN, N is set to MXN. See parameter MXN.

IHNK = Hankel function flag. If IHNK = 0, don't use Hankel function. If IHNK = 1, divide the starting field by the Hankel function, then multiply the solution field by the Hankel function before computing propagation loss. If starting field is Gaussian, IHNK should be set to 0. If starting field is elliptic, IHNK should be set to 1.

CARD	CONTENTS
	ITYPEB = type of bottom
	= 0, homogeneous Neumann boundary condition: program supplies bottom condition
	= 1, user supplies bottom condition. See subroutine BCON.
	= 2, artificial absorbing layer introduced: bottom of layer follows contour of water-bottom interface.
	= 3, artificial absorbing layer introduced: bottom of layer kept flat.
	ITYPES = type of surface
	= 0, pressure release: SCON sets SURY and SURX = 0.0.
	≠ 0, user inserts code in SCON to compute SURY and SURX.
2	RMAX, DR, WDR, WDZ, PDR, PDZ, ISFLD, ISVP, IBOT
where	RMAX = maximum range of solution (m).
	DR = range step for marching solution (m). If DR = 0, DR is set to 1/2 wavelength.
	WDR = range step (rounded to nearest DR) at which solution is written on disk (m).
	WDZ = depth increment (rounded to nearest DZ) at which solution is written on disk (m).
	PDR = range step (rounded to nearest DR) at which solution is printed (m).
	PDZ = depth increment (rounded to nearest DZ) at which solution is printed (m).
	ISFLD = 0, don't print starting field.
	= 1, print starting field.
	ISVP = 0, don't print sound speed profile.
	= 1, print sound speed profile.
	IBOT = 0, don't print bottom depths.
	1, print bottom depths.

CARD	CONTENTS		
3	R1, Z1		
4	R2, Z2	BOTTOM PROFILE	
5	R3, Z3	range and depth of water (m).	
.	. .	maximum number of depths = 100	
.	. .	(See parameter MXTRK.)	
N	. .		
N+1	-1, -1	Required. Marks end of bottom profile.	
N+2	RSVP		
N+3	KSVP		
N+4	NLYR		
N+5	ZLYR(I), RHO(I), BETA(I)		REPEAT
N+6	ZSVP(1), CSVP(1)	REPEAT	FOR EACH
N+7	ZSVP(2), CSVP(2)	FOR EACH	PROFILE
.	. .	LAYER.	
.	. .	I=1, NLYR	
N+M	ZSVP(J), CSVP(J)		
.	. .		

where

RSVP = range of SVP (m).

KSVP = SVP flag.

= 0, profile is in runstream.

≠ 0, profile (cards N+4 through N+M) is supplied by user-written subroutine USVP. KSVP may be used in computed GOTO statement to transfer control in user subroutine USVP.

NLYR = number of layers. If ITYPEB = 2 or 3, the program inserts an artificial absorbing layer and then increments NLYR by 1. Maximum NLYR = 100 (see parameter MXLYR).

ZLYR(I) = maximum depth of layer I in profile (m).

RHO(I) = density in layer I (g/cm^3).

BETA(I) = attenuation in layer I (dB/wavelength)

ZSVP = SVP depth (m).

Maximum number of sound speed profile values = 100 (see parameter MXSVP).

ZSVP(1) = depth to top of layer I (m).

ZSVP(J) = depth to bottom of layer I (m).

CSVP = SVP speed (m/s).

Maximum number of sound speed profile values = 100 (see parameter MXSVP).

CSVP(1) = speed of sound at top of layer I (m/s)

CSVP(J) = speed of sound at bottom of layer I (m/s).

3.3.2 Output Format

Output from the IFD model is written on disk in file IFD.OUT. File IFD.OUT is assigned to FORTRAN unit number NOU in the main program. The data written in IFD.OUT are unformatted and are written with FORTRAN WRITE statements as follows:

```
WRITE(NOU)FRQ,ZS,CO,ISF,RA,ZA,N,IHNK,ITYPEB,ITYPES,RMAX,DR,WDR,DZ,NLYR,ZLYR,RHO,BETA

WRITE(NOU)NZ,RA,WDZ,(U(I),I=IWZ,N,IWZ)
```

The first WRITE statement is executed only once and writes the value of each of the following parameters at the start of the problem.

FRQ = frequency (Hz)

ZS = source depth (m)

CO = reference sound speed (m/s)

ISF = starting field flag

= 0, Gaussian

= 1, user

RA = horizontal range from source to starting field (m).

ZA = depth of starting field (m).

N = number of equispaced receivers in the starting field.

IHNK = Hankel function flag.

= 0, Hankel function not used.

= 1, starting field was divided by Hankel function.

ITYPEB = type of bottom.

= 0, rigid.

= 1, user bottom.

= 2, artificial absorbing layer--follows bottom contour.

= 3, artificial absorbing layer--bottom kept flat.

ITYPES = type of surface

= 0, pressure release.

≠ 0, user supplied.

RMAX = maximum range of solution (m).

DR = range step for marching solution (m).

WDR = range step at which solution is written on disk (m).

DZ = depth increment of receivers (m).

NLYR = number of layers.

ZLYR = array containing depth of each layer (m).

RHO = array containing density in each layer (g/cm^3).

BETA = array containing attenuation in each layer (dB/wavelength).

The second WRITE statement is executed at each write-range increment, WDR. The data written are as follows:

NZ = number of equispaced receivers in the solution field.

RA = horizontal range from source to solution (m).

WDZ = depth increment at which solution is written on disk (m).

U = array that contains the complex field at range RA.

If IHNK=1, then the contents of U must be multiplied by the Hankel function before computing propagation loss.

IWZ = Index increment of receiver solutions to be written on disk.

The following READ statement may be used to read the solution field:

```
READ(unit) NZ,RA,WDZ,(U(I),I=1,NZ).
```

4. TEST PROBLEMS

The first test problem is an exact solution that has no physical significance other than to demonstrate the accuracy and flexibility of the IFD model. This problem requires the user to write the subroutines that supply the starting field, sound speed profiles, bottom condition, and surface condition.

Several other test problems that demonstrate the capability of the IFD model are also included. One problem deals with a range-independent environment; the others deal with range-dependent environments. Input runstreams and user-written subroutines that produced the IFD solutions to these problems are also included.

4.1 EXACT SOLUTION

As a test for accuracy, the IFD model was used to solve Burger's[13] kinetics-diffusion parabolic partial differential equation for which an exact solution is known. Burger's equation is

$$u_t = vu_{xx} - uu_x, \; 0 \leq x \leq 1, \; t \geq 0 \quad ,$$

where the subscripts denote partial derivatives. An exact solution for Burger's equation is

$$u(x,t) = \left\{ 1 + \exp \left[(x/2v) - (t/4v) \right] \right\}^{-1} \quad .$$

Substituting r for t and z for x and rewriting Burger's equation in the general form of equation (2.4), we have

$$u_r = (-u_z)u + vu_{zz} \; ,$$

where

$$(-u_z) = a(k_o, r,z) = \frac{ik_o(n^2 - 1)}{2}$$

$$v = b(k_o,r,z) = \frac{i}{2k_o} \quad .$$

Substituting r and z in the exact solution, we have

$$u(z,r) = \left(1 + e^{\frac{z}{2v} - \frac{r}{4v}} \right)^{-1} \quad .$$

The initial starting field and surface and bottom boundary conditions are then determined as follows:

Initial Starting Field

$$u(z,r) = \left(1 + e^{\frac{ik_o}{2}(-2z+r)}\right)^{-1} .$$

Surface Condition

$$u(o,r) = \left(1 + e^{\frac{irk_o}{2}}\right)^{-1} .$$

Bottom Condition

$$u(Zmax,r) = \left(1 + e^{\frac{ik_o}{2}(-2Zmax+r)}\right)^{-1} .$$

Solving for $(-u_z)$ gives

$$-u_z = -\frac{\partial u}{\partial z} = \frac{e^{\frac{2z-r}{4v}}}{2v\left(1 + e^{\frac{2z-r}{4v}}\right)^2} = n^2 + 1 ,$$

where $n^2 = \left(\frac{c_o}{c_i}\right)^2$,

c_o is the reference sound speed

c_i is the speed of sound at depth z.

Solving for c_i gives

$$c_i = c_o\sqrt{1 + \frac{1}{\cos\left(\frac{2\pi f}{c_o}(z - 0.5r)\right)}} .$$

Because of the constraints placed on the solution of Burger's equation, this test example has no real significance other than to test the accuracy

of the IFD model. One other important feature of this test example is that it requires the user to supply subroutines UFIELD, USVP, BCON, and SCON.

A comparison of the IFD and exact solutions at approximately 7 meters in range is shown in table 1.

Table 1. Comparison of IFD and Exact Solutions

I	Z(I)	U(I)
IFD 10 Exact	0.10	(0.49999E+00 -0.43163E+00) (0.50000E+00 -0.43165E+00)
20	0.20	(0.49998E+00 -0.41366E+00) (0.50000E+00 -0.41370E+00)
30	0.30	(0.49998E+00 -0.39631E+00) (0.50000E+00 -0.39635E+00)
40	0.40	(0.49997E+00 -0.37953E+00) (0.50000E+00 -0.37958E+00)
50	0.50	(0.49998E+00 -0.36328E+00) (0.50000E+00 -0.36333E+00)
60	0.60	(0.49998E+00 -0.34753E+00) (0.50000E+00 -0.34756E+00)
70	0.70	(0.49998E+00 -0.33222E+00) (0.50000E+00 -0.33225E+00)
80	0.80	(0.49999E+00 -0.31733E+00) (0.50000E+00 -0.31736E+00)
90	0.90	(0.49999E+00 -0.30285E+00) (0.50000E+00 -0.30286E+00)
100	1.00	(0.50000E+00 -0.28872E+00) (0.50000E+00 -0.28872E+00)

The input runstream and user-written subroutines UFIELD, USVP, BCON, and SCON are listed below.

Input Runstream for Exact Solution

```
100  .5  1500  1  0  1  100  0  1  1
7.1  .001  0  .1  1  .1  0  0  0
0  1
100  1
-1,-1
0
1
```

```
      SUBROUTINE UFIELD
C     ****************************************************************
C     *** USER STARTING FIELD
C     *** USER WRITES THIS SUBROUTINE IF GAUSSIAN FIELD NOT DESIRED
C     *** UFIELD IS CALLED IF INPUT PARAMETER ISF IS NOT ZERO
C     ****************************************************************
C     *** UFIELD SUBROUTINE SUPPLIES:
C         U   - COMPLEX STARTING FIELD
C     ****************************************************************
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C          U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C        BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C        ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C        RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C        X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
C     *** STARTING FIELD FOR EXACT SOLUTION TO BURGER'S PROBLEM
      DO 10 I=1,N
      ZI=I*DZ
      U(I)=1.0/(1.0+CEXP(CMPLX(0.0,.5*XKO*(-2.0*ZI+RA))))
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE USVP
C     *****************************************************************
C     *** USER SOUND VELOCITY PROFILE SUBROUTINE
C         SUBROUTINE USVP IS CALLED EACH DR IN RANGE AS LONG AS
C         KSVP IS NOT ZERO. KSVP MAY BE USED BY USER TO TRANSFER CONTROL
C         IN THIS SUBROUTINE. USER INSERTS LOGIC TO CLEAR KSVP
C         WHEN USVP IS NO LONGER NEEDED. IF KSVP NOT CLEARED BY USER,
C         USVP IS CALLED EACH STEP IN RANGE UNTIL RA = NEXT RSVP.
C     *****************************************************************
C     *** USVP SUBROUTINE RETURNS:
C         NLYR - NUMBER OF LAYERS. LAYER 1 IS WATER. OTHERS ARE SEDIMENT
C         ZLYR - ARRAY - DEPTH OF EACH LAYER. FIRST IS DEPTH OF WATER.
C         RHO  - ARRAY - DENSITY OF EACH LAYER. GRAMS/CUBIC CM
C         BETA - ARRAY - ATTENUATION IN EACH LAYER. DB/WAVELENGTH
C         IXSVP - ARRAY - CONTAINS POINTERS. POINTS TO LAST VALUE OF SVP
C                 IN CORRESPONDING LAYER. SVP IS STORED IN ARRAYS ZSVP
C                 AND CSVP. IXSVP(1) POINTS TO LAST SVP POINT IN WATER.
C         NSVP - NUMBER OF POINTS IN ZSVP AND CSVP. ZSVP AND CSVP
C                 CONTAIN THE PROFILES FOR ALL LAYERS.
C         ZSVP - ARRAY - SVP DEPTHS - METERS
C         CSVP - ARRAY - SOUND SPEED - METERS/SEC
C         KSVP - AS DESCRIBED ABOVE.
C     *****************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),C0,CSVP(MXSVP),DR,DR1,DZ,FRO,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XK0,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      GO TO (100,200,300,400) ,KSVP
      NSVP=0
      RETURN
C
100   CONTINUE
C
C     *** IF KSVP=1, CONTROL IS TRANSFERRED HERE. USER LOADS
C         NLYR,ZLYR(I),RHO(I),BETA(I), AND IXSVP(I) WHERE I=1,NLYR.
C         USER ALSO LOADS NSVP,ZSVP(I), AND CSVP(I) WHERE I=1,NSVP.
C         KSVP MAY BE ALTERED DEPENDING ON USER LOGIC.
C
C     *** SVP FOR EXACT SOLUTION
C
      NLYR=1
      ZLYR(1)=1.0
      RHO(1)=1.0
      BETA(1)=0.0
      NSVP=101
      DZSVP=ZLYR(1)/(NSVP-1)
      XK0=2.0*PI*FRO/C0
      DO 110 I=1,NSVP
```

```
      ZI=(I-1)*DZSVP
      CSVP(I)=CO*SQRT(1.0+1.0/(COS(XKO*(ZI-.5*RA))))
      ZSVP(I)=ZI
110   CONTINUE
      IXSVP(1)=NSVP
      RETURN
C
200   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
C
300   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
C
400   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
      END
```

```
      SUBROUTINE BCON
C     ******************************************************************
C     *** USER PREPARED BOTTOM CONDITION SUBROUTINE
C         BCON IS CALLED IF INPUT PARAMETER ITYPEB = 1
C         SEE MAIN PROGRAM FOR DEFINITIONS
C     ******************************************************************
C     *** SUBROUTINE RETURNS:
C         BOTY,BOTX
C     ******************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      IF(THETA) 50,100,150
C
C     *** THETA LESS THAN 0.0. BOTTOM SLOPES UP.
50    CONTINUE
      BOTY=U(N)
C     BOTX=..............
      RETURN
C
C     *** THETA = 0.0. BOTTOM IS FLAT.
C
C     *** BOTTOM CONDITION FOR EXACT SOLUTION TO BURGER'S PROBLEM
100   CONTINUE
      BOTY=U(N)
      BOTX=1.0/(1.0+CEXP(CMPLX(0.0,.5*XKO*(-2.0*ZA+RA))))
      RETURN
C
C     *** THETA GREATER THAN 0.0, BOTTOM SLOPES DOWN.
150   CONTINUE
C     BOTY=..............
C     BOTX=..............
      RETURN
      END
```

```
      SUBROUTINE SCON
C     ******************************************************************
C     *** SURFACE CONDITION SUBROUTINE
C         IF ITYPES = 0, SCON SETS SURY AND SURX = 0.0.
C         IF ITYPES NOT 0, THE USER MUST SUPPLY SURY AND SURX.
C         SEE MAIN PROGRAM FOR DEFINITIONS
C     ******************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      IF(ITYPES.NE.0) GO TO 100
C
C     *** PRESSURE RELEASE SURFACE
      SURY=0.0
      SURX=0.0
      RETURN
C
C     *** USER SURFACE CONDITION
C     *** SURFACE CONDITION FOR EXACT SOLUTION TO BURGER'S PROBLEM
100   CONTINUE
      SURY=1.0/(1.0+CEXP(CMPLX(0.0,.5*XKO*(RA-DR))))
      SURX=1.0/(1.0+CEXP(CMPLX(0.0,.5*XKO*(+RA))))
      RETURN
      END
```

4.2 RANGE-INDEPENDENT PROBLEMS

The selection of range-independent problems includes treatment of an isovelocity shallow water environment and a horizontal interface.

4.2.1 Isovelocity Shallow Water

This problem, published by Jensen and Kuperman,[14] considers a simple isovelocity shallow water environment. The sound speed in the water is 1500 m/s. The water depth is 100 m, and both source and receiver are placed 50 m in depth. In the bottom, the sound speed is 1550 m/s, density is 1.2 g/cm^3, and the attenuation is 1 dB/wavelength. The source frequency is 500 Hz. The propagation path is up to 25 km in range.

Solutions obtained by Jensen and Kuperman, using a normal mode model (SNAP) and a parabolic equation split-step model (PAREQ) developed at SACLANT Centre, are compared with the solution obtained with the IFD model. As shown in figure 5, all solutions are in excellent agreement.

The input IFD runstream that produced these results is listed below.

Input Runstream

```
500  50  0  0  0  250  500  0  3  0
25000  5  50  50  5000  50  0  0  0
0  100
25000  100
-1,-1
0
0
2
100  1.0  -1.0
0  1500
100  1500
200  1.2  1.0
100  1550
200  1550
```

Note that although the bottom parameters were extended down to 200 m, the maximum depth of the solution was extended to 250 m as requested by the combination of input parameters ZA and ITYPEB. Artificial attenuation was then applied to the bottom-most 50 m as described by Brock.[10]

4.2.2 Horizontal Interface

This problem, suggested by Dr. H. Bucker of NOSC in a personal communication, considers propagation in a region where the sound speed profile is as depicted in figure 6. Source and receiver depths are 30 m and 90 m, respectively. Source frequency is 100 Hz. At 240 m in depth, the density changes abruptly from 1.0 g/cm^3 to 2.1 g/cm^3. No attenuation is applied.

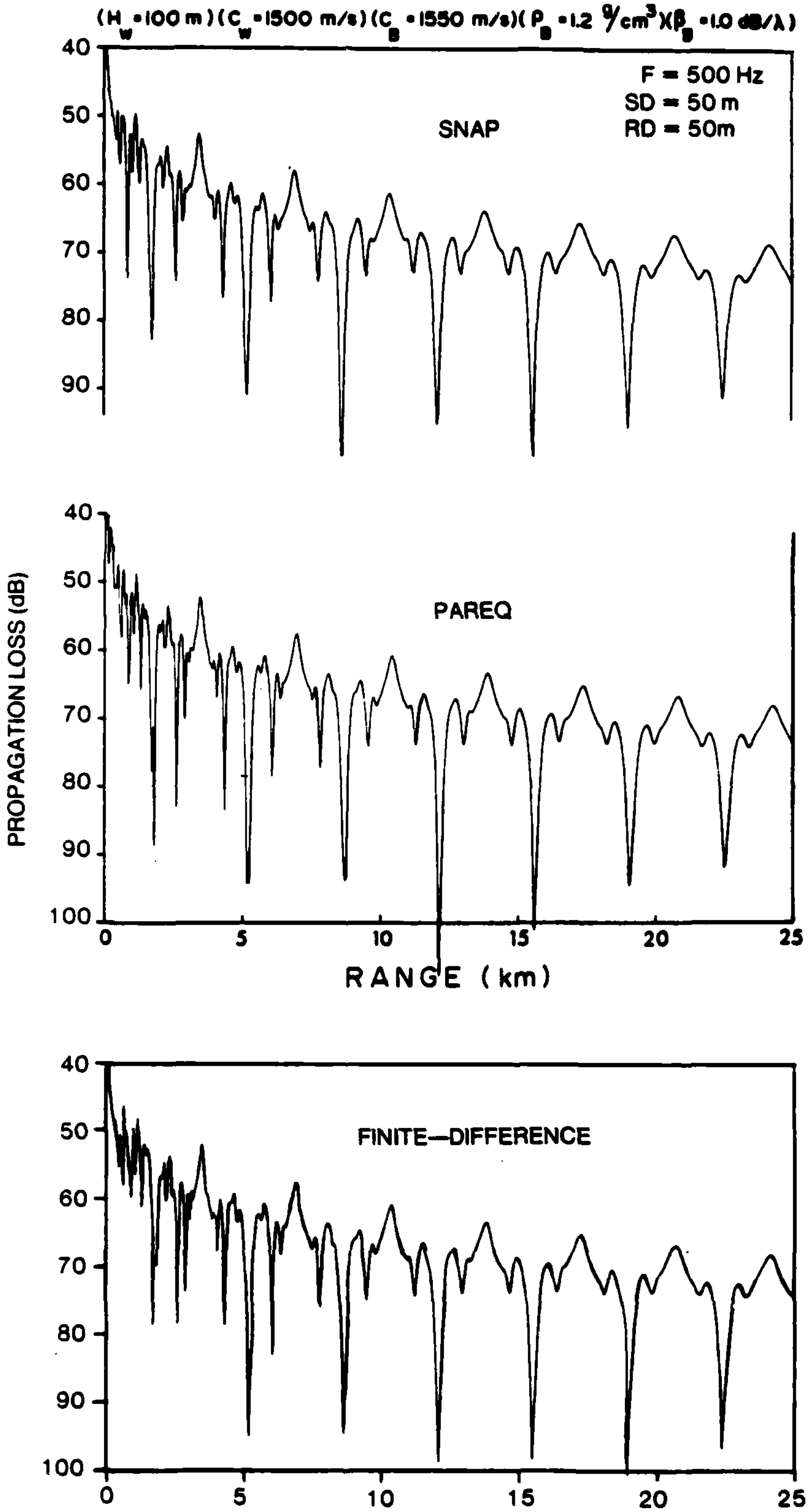


Figure 5. Propagation Loss Versus Range for Shallow Water Propagation

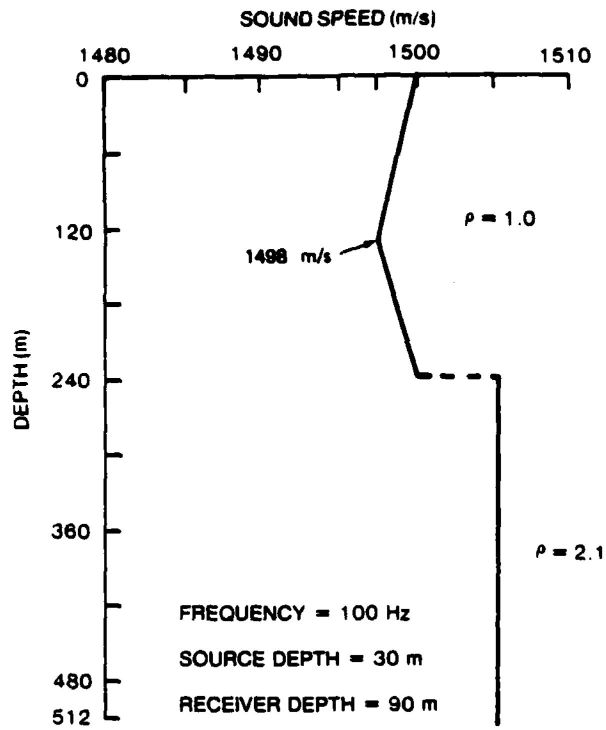


Figure 6. Horizontal Interface Problem

The solution produced by the IFD model and a normal mode solution provided by Dr. Bucker are compared in figure 7.

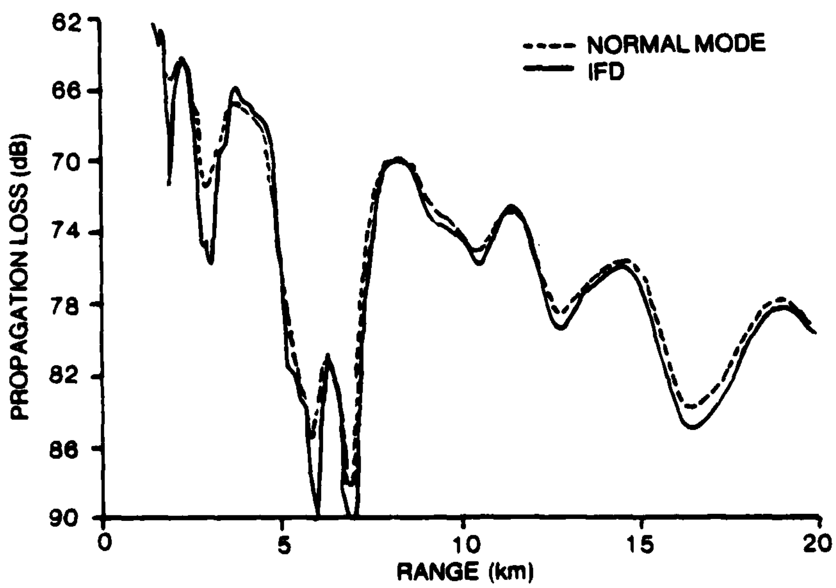


Figure 7. Solution of Horizontal Interface Problem

The input runstream that produced these results is listed below. It should be noted that the bottom was artificially extended to 1200 m.

<u>Input Runstream for Horizontal Interface Problem</u>

```
100 30 0 0 0 1200 600 0 3 0
20000 2 50 90 10000 50 0 0 0
0 240
20000 240
-1,-1
0
0
2
240 1.0 0.0
0 1500
120 1498
240 1500
512 2.1 0.0
240 1505
512 1505
```

4.3 RANGE-DEPENDENT PROBLEMS

The selection of *range-dependent* problems includes treatment of depth dependent environments, interface conditions, and a homogeneous Neumann bottom boundary condition.

4.3.1 <u>Shallow-to-Deep Water Problem</u>

This problem, extracted from Jensen and Kuperman,[14] considers propagation in shallow-to-deep water as shown in figure 8. The region of propagation is bounded by a pressure release surface and an irregular bottom where the bottom remains flat at 50 m in depth for the first 10 km; at 10 km the bottom begins to slope downward until it levels off and remains flat at 350 m in depth. Propagation loss was calculated at two different sloping angles, 0.85 and 8.5 degrees. The sound speed in the water is 1500 m/s; in the bottom it is 1600 m/s. In the bottom, a density of 1.5 g/cm^3 and an attenuation of 0.2 dB/wavelength are used. The source frequency is 25 Hz, and the source and receiver are both placed at a depth of 25 m.

The results produced by SNAP, PAREQ, and the IFD model are shown in figures 9 and 10.

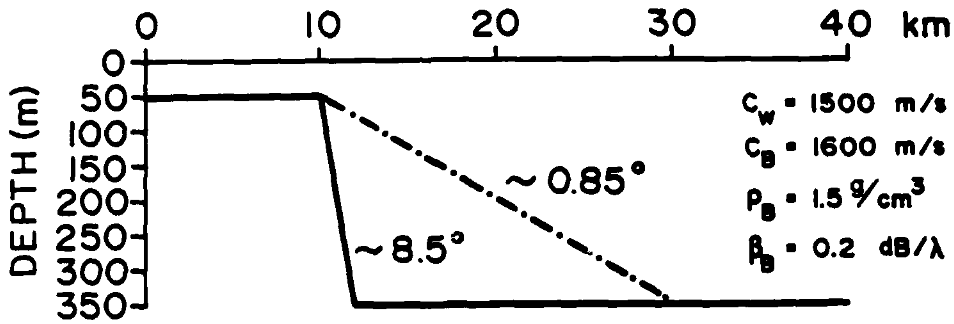


Figure 8. Shallow-to-Deep Water Propagation

The input runstream that produced the IFD results for the 0.85 degree sloping bottom is listed below.

Input Runstream (0.85 degree slope)

```
25 25 1500 0 0 1000 1000 0 3 0
40000 10 100 25 10000 25 0 0 0
0 50
10000 50
30000 350
40000 350
-1,-1
0
0
2
50 1.0 -1.0
0 1500
50 1500
750 1.0 .2
50 1600
750 1600
10000
1
30000
0
2
350 1.0 -1.0
0 1500
350 1500
750 1.0 .2
350 1600
750 1600
```

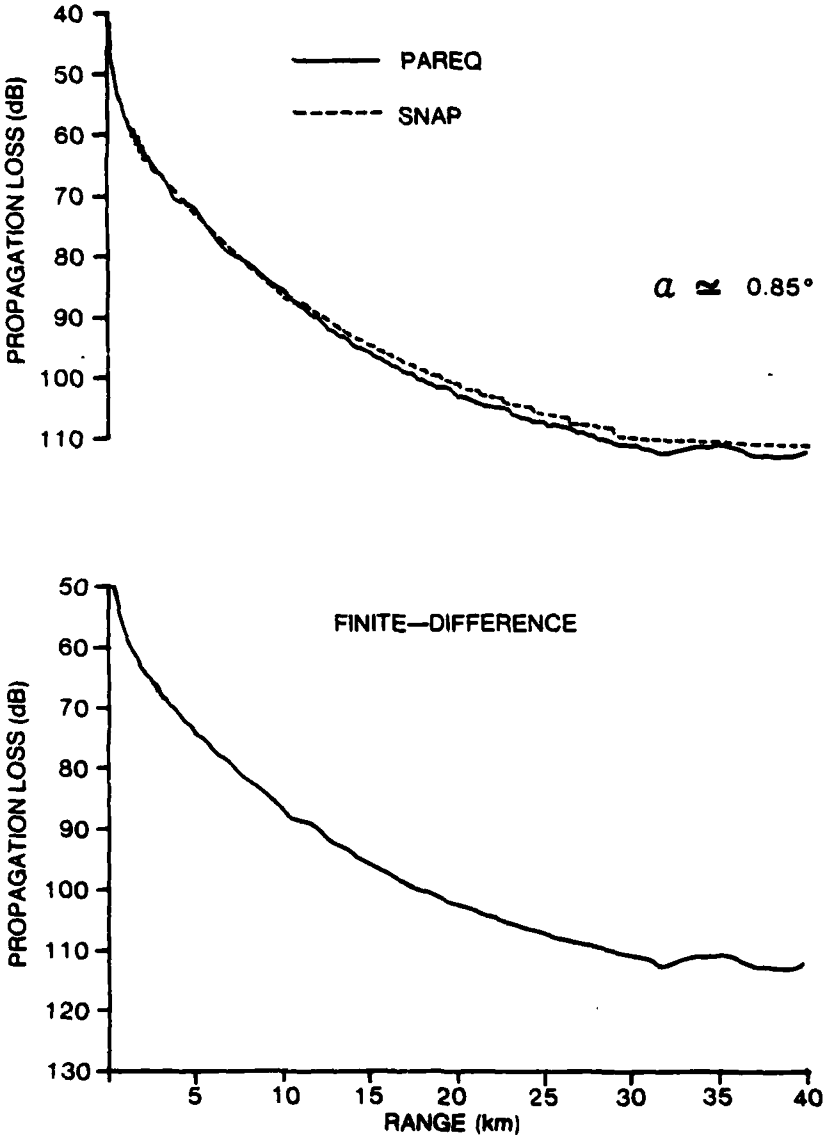


Figure 9. Propagation Loss Versus Range for Shallow-to-Deep Water Propagation, 0.85 degree Slope

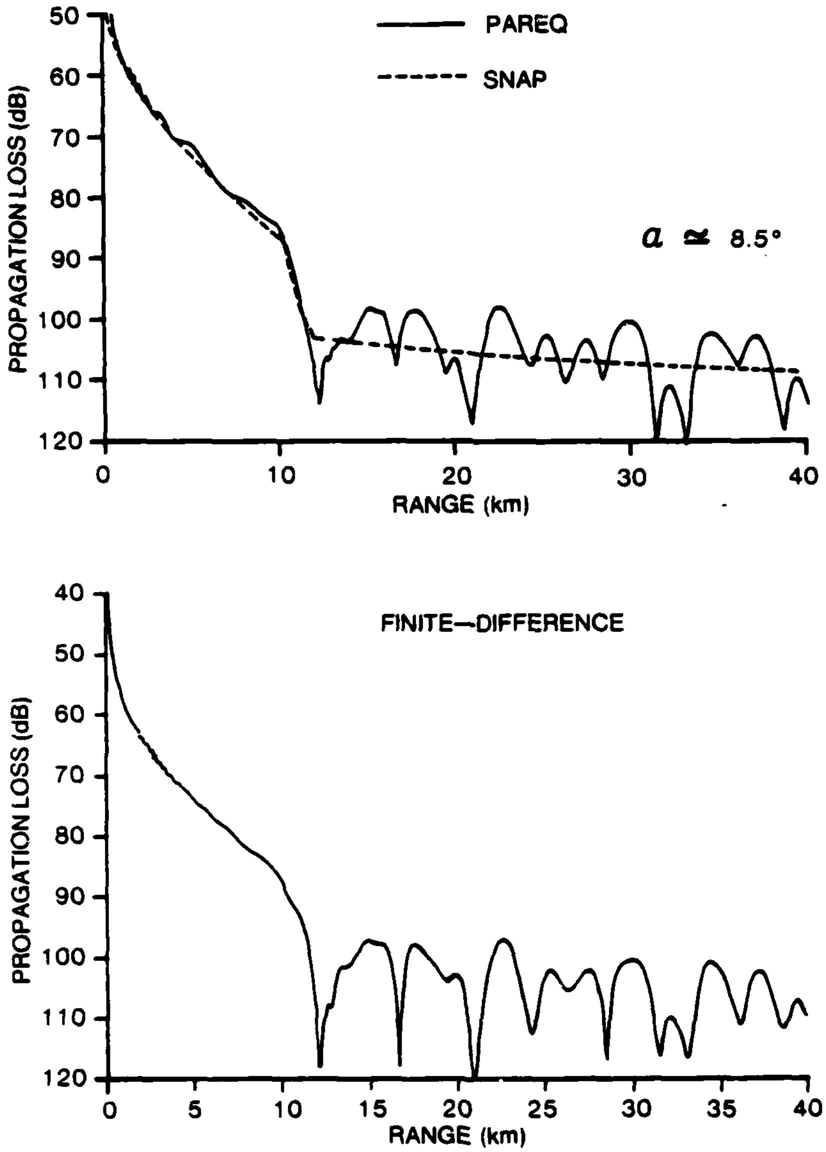


Figure 10. Propagation Loss Versus Range for Shallow-to-Deep Water Propagation, 8.5 degree Slope

As requested in the input runstream, user subroutine USVP is called to supply sound speed profiles over the region from 10 to 30 km in range. Subroutine USVP is included below.

```
      SUBROUTINE USVP
C     *****************************************************************
C     *** USER SOUND VELOCITY PROFILE SUBROUTINE
C         SUBROUTINE USVP IS CALLED EACH DR IN RANGE AS LONG AS
C         KSVP IS NOT ZERO. KSVP MAY BE USED BY USER TO TRANSFER CONTROL
C         IN THIS SUBROUTINE. USER INSERTS LOGIC TO CLEAR KSVP
C         WHEN USVP IS NO LONGER NEEDED. IF KSVP NOT CLEARED BY USER,
C         USVP IS CALLED EACH STEP IN RANGE UNTIL RA = NEXT RSVP.
C     *****************************************************************
C     *** USVP SUBROUTINE RETURNS:
C         NLYR - NUMBER OF LAYERS. LAYER 1 IS WATER. OTHERS ARE SEDIMENT
C         ZLYR - ARRAY - DEPTH OF EACH LAYER. FIRST IS DEPTH OF WATER.
C         RHO  - ARRAY - DENSITY OF EACH LAYER. GRAMS/CUBIC CM
C         BETA - ARRAY - ATTENUATION IN EACH LAYER. DB/WAVELENGTH
C         IXSVP - ARRAY - CONTAINS POINTERS. POINTS TO LAST VALUE OF SVP
C                IN CORRESPONDING LAYER. SVP IS STORED IN ARRAYS ZSVP
C                AND CSVP. IXSVP(1) POINTS TO LAST SVP POINT IN WATER.
C         NSVP - NUMBER OF POINTS IN ZSVP AND CSVP. ZSVP AND CSVP
C                CONTAIN THE PROFILES FOR ALL LAYERS.
C         ZSVP - ARRAY - SVP DEPTHS - METERS
C         CSVP - ARRAY - SOUND SPEED - METERS/SEC
C         KSVP - AS DESCRIBED ABOVE.
C     *****************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      GO TO (100,200,300,400) ,KSVP
      NSVP=0
      RETURN
C
100   CONTINUE
C
C     *** IF KSVP=1, CONTROL IS TRANSFERRED HERE. USER LOADS
C         NLYR,ZLYR(I),RHO(I),BETA(I), AND IXSVP(I) WHERE I=1,NLYR.
C         USER ALSO LOADS NSVP,ZSVP(I), AND CSVP(I) WHERE I=1,NSVP.
C         KSVP MAY BE ALTERED DEPENDING ON USER LOGIC.
C
C     *** USER SUPPLIES SVP
C
      NLYR=2
      ZLYR(1)=50.0+(RA-10000.0)*TAN(THETA)
      RHO(1)=1.0
      BETA(1)=-1.0
      ZSVP(1)=0.0
```

```
      CSVP(1)=1500.0
      ZSVP(2)=ZLYR(1)
      CSVP(2)=1500.0
      IXSVP(1)=2
      ZLYR(2)=750.0
      RHO(2)=1.0
      BETA(2)=.2
      ZSVP(3)=ZSVP(2)
      CSVP(3)=1600.0
      ZSVP(4)=750.0
      CSVP(4)=1600.0
      NSVP=4
      IXSVP(2)=4
      RETURN
C
200   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
C
300   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
C
400   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
      END
```

4.3.2 Deep-to-Shallow Water Problem

This problem, also extracted from Jensen and Kuperman,[14] considers propagation in deep-to-shallow water as shown in figure 11. The environment is identical to that of the previous problem except that the shallow and deep portions are reversed such that the bottom slopes upward in the direction of propagation. The results produced by SNAP, PAREQ, and IFD are shown in figures 12 and 13.

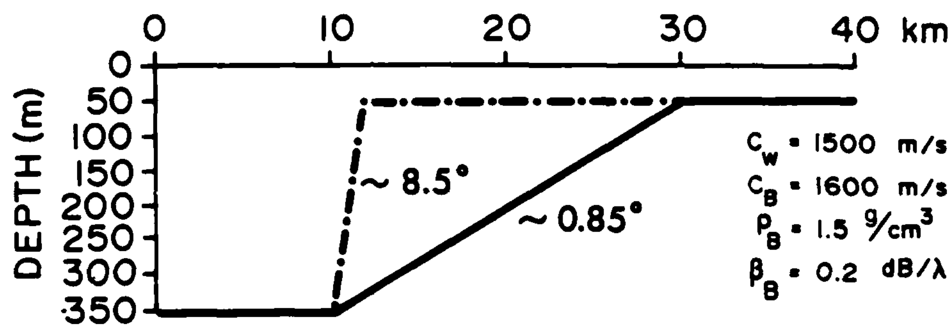


Figure 11. Deep-to-Shallow Water Propagation

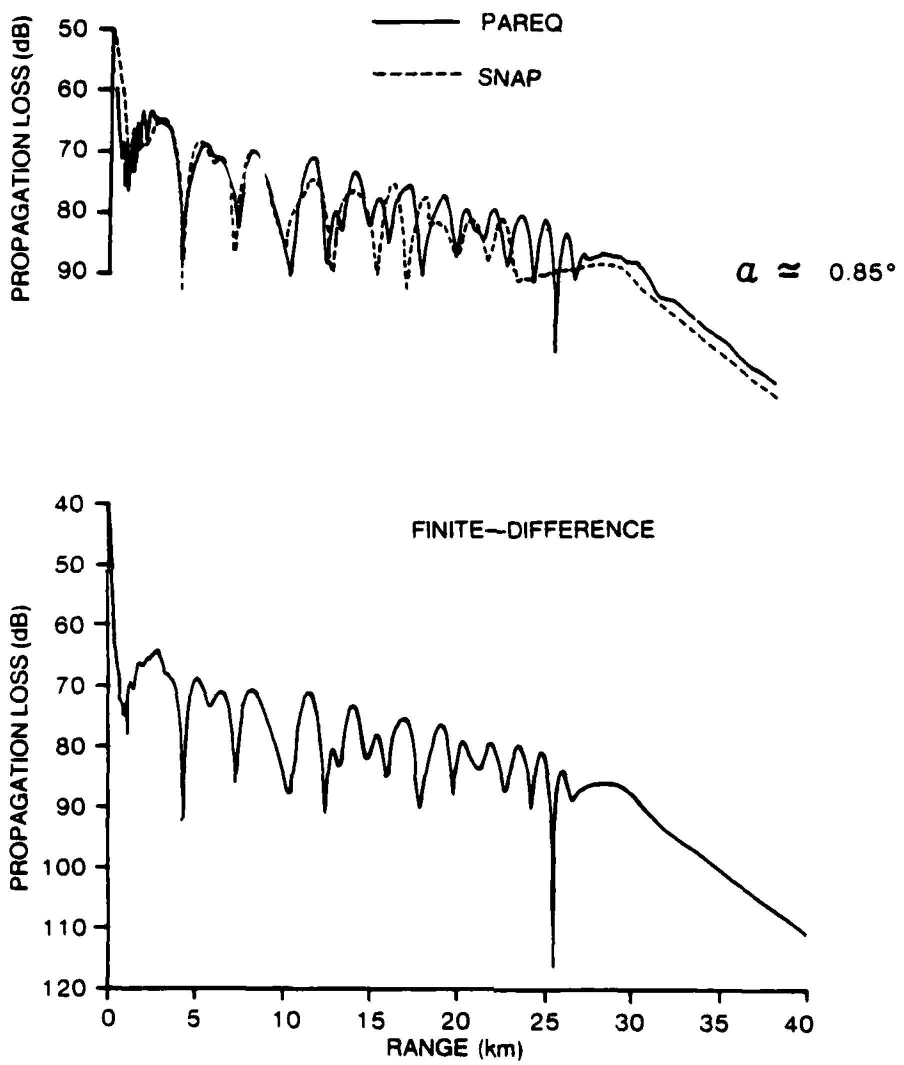


Figure 12. Propagation Loss Versus Range for Deep-to-Shallow Water Propagation, 0.85 degree Slope

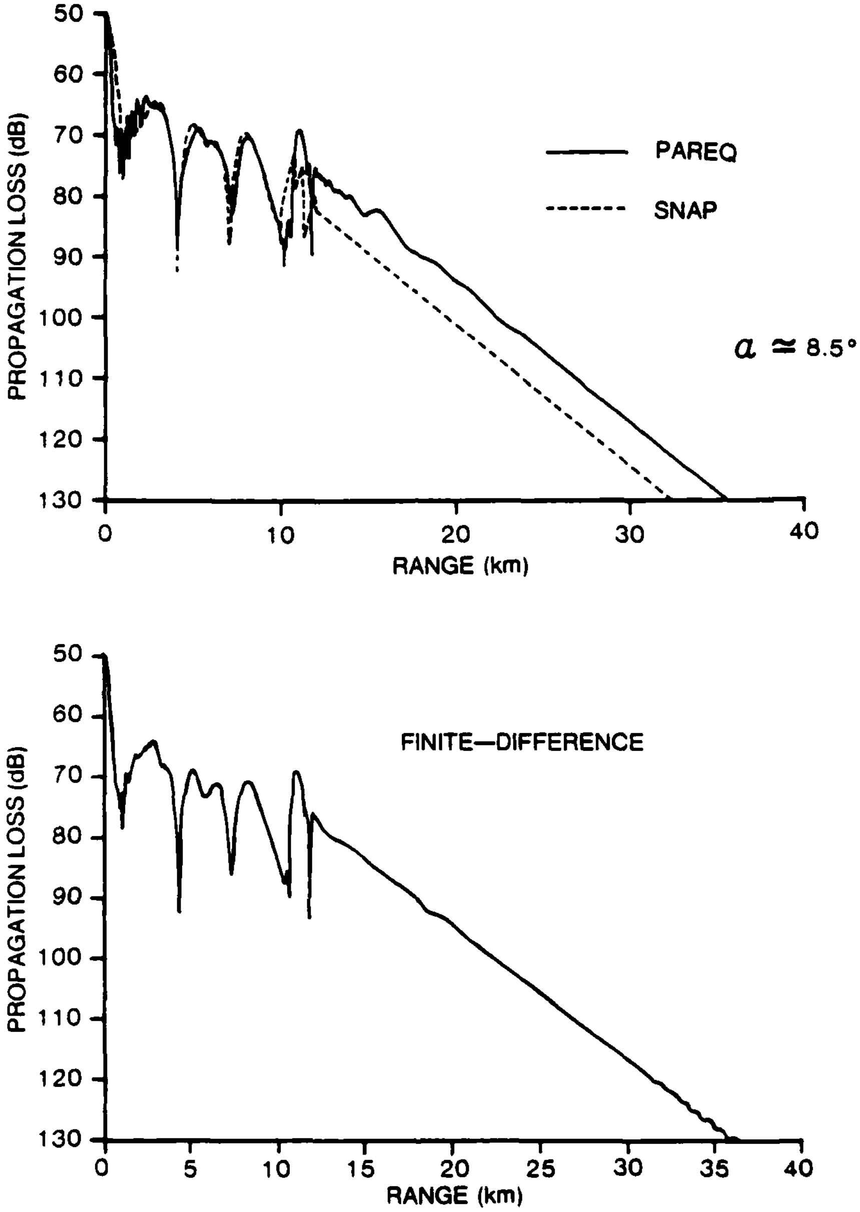


Figure 13. Propagation Loss Versus Range for Deep-to-Shallow Water Propagation, 8.5 degree Slope

The input runstream and user subroutine USVP which produced the IFD results for the 8.5 degree slope are listed below.

Input Runstream (8.5 degree slope)

```
25 25 0 0 0 1000 1000 0 3 0
40000 10 100 25 10000 25 0 0 0
0 350
10000 350
12000 50
40000 50
-1,-1
0
0
2
350 1.0 -1.0
0 1500
350 1500
750 1.0 .2
350 1600
750 1600
10000
1
12000
0
2
50 1.0 -1.0
0 1500
50 1500
750 1.0 .2
50 1600
750 1600
```

```
      SUBROUTINE USVP
C     ****************************************************************
C     *** USER SOUND VELOCITY PROFILE SUBROUTINE
C         SUBROUTINE USVP IS CALLED EACH DR IN RANGE AS LONG AS
C         KSVP IS NOT ZERO. KSVP MAY BE USED BY USER TO TRANSFER CONTROL
C         IN THIS SUBROUTINE. USER INSERTS LOGIC TO CLEAR KSVP
C         WHEN USVP IS NO LONGER NEEDED. IF KSVP NOT CLEARED BY USER,
C         USVP IS CALLED EACH STEP IN RANGE UNTIL RA = NEXT RSVP.
C     ****************************************************************
C     *** USVP SUBROUTINE RETURNS:
C         NLYR - NUMBER OF LAYERS. LAYER 1 IS WATER. OTHERS ARE SEDIMENT
C         ZLYR - ARRAY - DEPTH OF EACH LAYER. FIRST IS DEPTH OF WATER.
C         RHO  - ARRAY - DENSITY OF EACH LAYER. GRAMS/CUBIC CM
C         BETA - ARRAY - ATTENUATION IN EACH LAYER. DB/WAVELENGTH
C         IXSVP - ARRAY - CONTAINS POINTERS. POINTS TO LAST VALUE OF SVP
C                IN CORRESPONDING LAYER. SVP IS STORED IN ARRAYS ZSVP
C                AND CSVP. IXSVP(1) POINTS TO LAST SVP POINT IN WATER.
C         NSVP - NUMBER OF POINTS IN ZSVP AND CSVP. ZSVP AND CSVP
C                CONTAIN THE PROFILES FOR ALL LAYERS.
C         ZSVP - ARRAY - SVP DEPTHS - METERS
C         CSVP - ARRAY - SOUND SPEED - METERS/SEC
C         KSVP - AS DESCRIBED ABOVE.
C     ****************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      GO TO (100,200,300,400) ,KSVP
      NSVP=0
      RETURN
C
100   CONTINUE
C
C     *** IF KSVP=1, CONTROL IS TRANSFERRED HERE. USER LOADS
C         NLYR,ZLYR(I),RHO(I),BETA(I), AND IXSVP(I) WHERE I=1,NLYR.
C         USER ALSO LOADS NSVP,ZSVP(I), AND CSVP(I) WHERE I=1,NSVP.
C         KSVP MAY BE ALTERED DEPENDING ON USER LOGIC.
C
C     *** USER SUPPLIES SVP
C
      NLYR=2
      ZLYR(1)=350.0+(RA-10000.0)*TAN(THETA)
      RHO(1)=1.0
      BETA(1)=-1.0
      ZSVP(1)=0.0
```

```
      CSVP(1)=1500.0
      ZSVP(2)=ZLYR(1)
      CSVP(2)=1500.0
      IXSVP(1)=2
      ZLYR(2)=750.0
      RHO(2)=1.0
      BETA(2)=.2
      ZSVP(3)=ZSVP(2)
      CSVP(3)=1600.0
      ZSVP(4)=750.0
      CSVP(4)=1600.0
      NSVP=4
      IXSVP(2)=4
      RETURN
C
200   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
C
300   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
C
400   CONTINUE
C     *** USER INSERTS CODE HERE IF DESIRED
      RETURN
      END
```

4.3.3 Shallow-to-Deep Water Propagation in a Wedge-Shaped Region With A Rigid Sloping Bottom

In this example, the IFD model was used to propagate the acoustic field in a wedge-shaped region with a rigid sloping bottom as shown in figure 14. The purpose of this example is to exercise the optional rigid bottom boundary condition programmed in the model. For this case, the source frequency is 80 Hz; source depth is 15.2 m; bottom depth at the location of the source is 30.5 m; and the sound speed profile is constant at 1524 m/s. The bottom is rigid and slopes downward at an angle of 5 degrees. The initial field propagated by the IFD model was generated by the method of images[15] and is at an initial range of 348.4 m from the source.

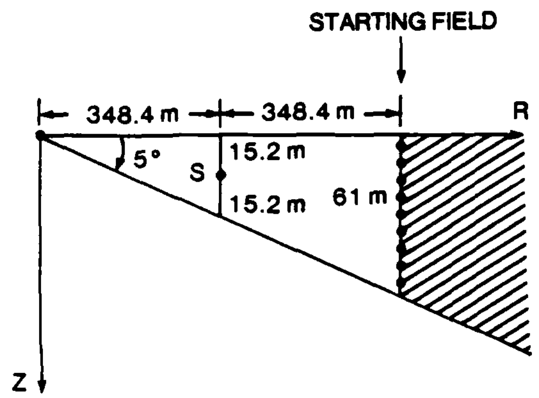


Figure 14. Shallow-to-Deep Water Propagation, Wedge-Shaped Region With a Rigid Sloping Bottom

Numerical results were compared with the exact solution obtained by the method of images. A plot of propagation loss versus range at a receiver depth of 27.4 m is given in figure 15.

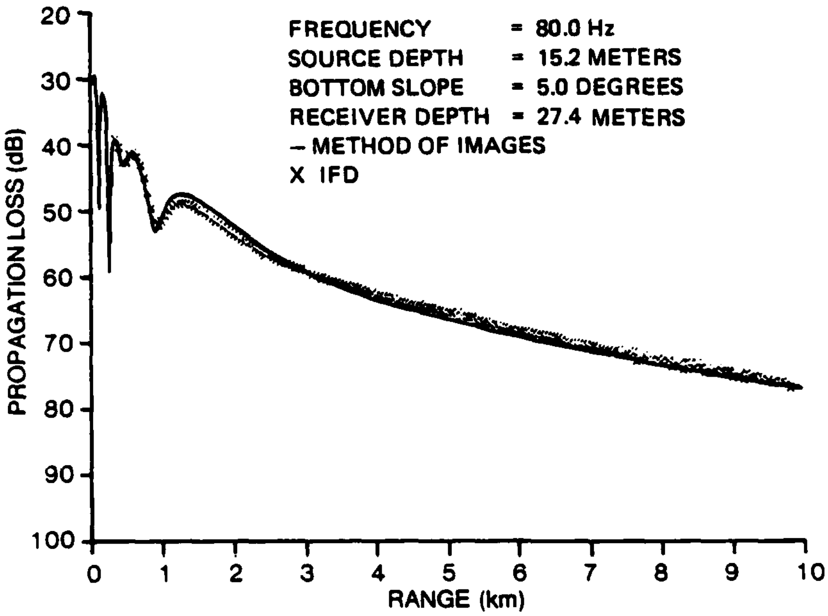


Figure 15. Propagation Loss Versus Range, Wedge-Shaped Region With a Rigid Sloping Bottom

The initial field generated by the method images consisted of 400 points spaced at approximately 0.15 m in depth. As the solution was marched out in range, the field was extended deeper and deeper until, at 10 km, the field consisted of 5740 points in depth.

The input runstream and user subroutine UFIELD which produced these results are included below.

<u>Input Runstream</u>

```
80 15.24003 0 1 348.3886 60.96012 400 1 0 0
10000 10 50 .5 5000 27 0 0 0
0 30.48006
10000 905.38
-1,-1
0
0
1
30.48006 1.0 0.0
0 1524.003
30.48006 1524.003
```

```
      SUBROUTINE UFIELD
C     ****************************************************************
C     *** USER STARTING FIELD
C     *** USER WRITES THIS SUBROUTINE IF GAUSSIAN FIELD NOT DESIRED
C     *** UFIELD IS CALLED IF INPUT PARAMETER ISF IS NOT ZERO
C     ****************************************************************
C     *** UFIELD SUBROUTINE SUPPLIES:
C         U   - COMPLEX STARTING FIELD
C     ****************************************************************
      PARAMETER NUU=3
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
C     *** STARTING FIELD GENERATED BY WEDGE PROGRAM. CONSTANT SVP.
C         MUST BE DIVIDED BY HANKEL FUNCTION.
C         SET IHNK = 1 IN IFD INPUT RUNSTREAM.
C
      CALL ASSIGN(NUU,'WEDGE2.FLD')
C
C     *** BYPASS WEDGE DATA
      READ(NUU) NANG,F,C1,ZSA,ZSBB,RMIN,RMAX,DRR,ZMIN,ZMAX,DZZ,PHI
C
C     *** READ WEDGE STARTING FIELD
      READ(NUU) NZ,R,(U(I),I=1,NZ)
C     *** NZ IS NUMBER OF DEPTHS. - SHOULD BE EQUAL TO N
C     *** R IS RANGE IN FT. - SHOULD BE EQUAL TO RA IN METERS
C
C     *** WEDGE REFERENCES TO 1 METER BY ADDING -20.0*ALOG10(3.280833)
C     *** PROGRAM WHICH PLOTS IFD SOLUTION SHOULD DO SAME.
C
      CALL CLOSE(NUU)
      RETURN
      END
```

5. CONCLUSIONS

The IFD model exhibits significant advantages for solving the parabolic wave equation because of its generality, useful capabilities, unconditional stability of the method, and efficient computation of the wave field. In the present design, the program package is basic in structure and is flexible enough for easy generalization and modification.

The capabilities for handling surface and bottom boundary conditions as well as horizontal interface conditions are important features which enhance the parabolic equation model. Further capabilities can be incorporated without much difficulty.

The solution of the system (equation (2.21)) by a sparse matrix technique is economical and efficient. The disadvantage of the IFD method is that the boundary condition information at the next range-level must be known. In cases where the bottom condition is unknown, the option to extend the bottom with an artificial absorbing layer such that the field at the bottom becomes zero may be helpful.

This report presented the status of the IFD model as of this writing. Future capabilities to be built into the model include wide angle propagation, multiple irregular interfaces, automatic step-size determination, high frequency propagation, shear waves, and others.

6. REFERENCES

1. F. D. Tappert and R. H. Hardin, "Computer Simulation of Long-Range Ocean Acoustic Propagation Using the Parabolic Equation Method," Proceedings of the Eighth International Congress on Acoustics (London, 1974), vol. 2, p. 452.

2. D. Lee, G. Botseas, and J. S. Papadakis, "Finite-Difference Solution to the Parabolic Wave Equation," Journal of the Acoustical Society of America, vol. 70, no. 3, 1981, pp. 795-800.

3. D. Lee and J. S. Papadakis, Numerical Solutions of Underwater Acoustic Wave Propagation Problems, NUSC Technical Report 5929, 1979.

4. F. D. Tappert, "The Parabolic Approximation Method," in Wave Propagation and Underwater Acoustics, ed. by J. B. Keller and J. S. Papadakis, Lecture Notes in Physics, vol. 70, Springer-Verlag, Heidelberg, 1977.

5. P. R. Garabedian, Partial Differential Equations, John Wiley & Sons, New York, 1964.

6. A. R. Mitchell, Computational Methods in Partial Differential Equations, John Wiley & Sons, New York, 1976.

7. D. Lee and S. Preiser, "Generalized Adams Methods for Solving Underwater Wave Propagation Problems," Journal of Computers and Mathematics With Applications, vol. 7, no. 2, 1981, pp. 195-202.

8. S. T. McDaniel and D. Lee, "A Finite-Difference Treatment of Interface Conditions for the Parabolic Wave Equation: The Horizontal Interface," to appear in Journal of the Acoustical Society of America.

9. F. Jensen and H. Krol, "The Use of Parabolic Equation Method in Sound Propagation Modeling," SACLANTCEN memorandum SM-72, 1975.

10. H. K. Brock, "The AESD Parabolic Equation Model," NORDA Technical Note 12, 1978.

11. B. Carnahan, H. A. Luther, and J. O. Wikes, Applied Numerical Methods, John Wiley & Sons, Inc., 1969.

12. M. Abramowitz and I. Stegun (eds.), Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables, Applied Mathematics, Series 55, U.S. Govt. Printing Office, Washington, D.C., 1964.

13. G. D. Byrne et al., "A Comparison of Two ODE Codes: GEAR and EPISODE," Journal of Computers and Chemical Engineering, vol. 1, 1977, pp. 133-147.

14. F. B. Jensen and W. A. Kuperman, Environmental Acoustical Modeling at SACLANTCEN, SACLANTCEN Report SR-34, 1979.

15. R. B. Lauer, R. L. Deavenport, and D. M. Potter, "The Acoustic Field in a Homogeneous Wedge as Given by the Method of Images," NUSC Technical Memorandum TA11-88-75, 1975.

Appendix A

IFD COMPUTER LISTING

```
C      ****************************************************************
C      ****************************************************************
C      *  IMPLICIT FINITE-DIFFERENCE METHOD FOR SOLVING THE           *
C      *  PARABOLIC EQUATION :  U =A*U+B*U     ; WHERE                *
C      *                         R       ZZ                           *
C      *                        A=I*.5*KO*(N*N-1) ; AND B=I*.5/KO     *
C      ****************************************************************
C      ****************************************************************
C      *  D. LEE AND G. BOTSEAS, CODE 3342                            *
C      *  NAVAL UNDERWATER SYSTEMS CENTER                             *
C      *  NEW LONDON, CONNECTICUT 06320, U.S.A.                       *
C      ****************************************************************
C      ****************************************************************
C      *  VAX-11/780 VERSION - FORTRAN IV+                            *
C      ****************************************************************
C      *** ACOFX - COEFFICIENT 'A' AT BOTTOM - AT ADVANCED RANGE
C      *** ACOFY - COEFFICIENT 'A' AT BOTTOM - AT PRESENT RANGE
C      *** ALPHA - VOLUME ATTENUATION - DB/METER
C      *** ATT   - ATTENUATION COEFFICIENT FOR ARTIFICIAL ABSORBING LAYER
C      *** BCOF  - COEFFICIENT 'B'  - RANGE INDEPENDENT
C      *** BETA  - ARRAY - ATTENUATION IN LAYERS - DB/WAVELENGTH
C      *** BOTX  - COMPLEX PRESSURE AT BOTTOM AT ADVANCED RANGE RA+DR
C      *** BOTY  - COMPLEX PRESSURE AT BOTTOM AT PRESENT RANGE RA
C      *** BTA   - ARRAY - PARTIAL SOLUTION OF SYSTEM OF EQUATIONS
C      *** CO    - REFERENCE SPEED OF SOUND - METERS/SEC
C      *** CSVP  - ARRAY - SOUND VELOCITY - METERS/SEC
C      *** DEG   - CONVERSION FACTOR - DEGREES/RADIAN
C      *** DR    - RANGE STEP - METERS
C      *** DR1   - LAST DR USED IN ROUTINE DIAG
C      *** DZ    - DEPTH INCREMENT OF SOLUTION - METERS
C      *** DZZ   - DEPTH INCREMENT FOR ADJUSTING LAYER DEPTHS IN SLOPING BOTTOM
C      *** FRQ   - FREQUENCY - HZ
C      *** HNK   - HANKEL FUNCTION HO(1)
C      *** HNKL  - EXTERNAL FUNCTION - COMPUTES HANKEL FUNCTION HO(1)
C      *** IBOT  - BOTTOM DEPTH PRINT FLAG
C                  = 0 - DO NOT PRINT BOTTOM DEPTHS
C                  = 1 - PRINT BOTTOM DEPTHS
C      *** IDIAG - DIAGONAL UPDATE FLAG
C      *** IHNK  - HANKEL FUNCTION FLAG
C                  = 0 - HANKEL FUNCTION NOT USED. 10*LOG(R) ADDED TO
C                        SOLUTION.
C                  = 1 - STARTING FIELD DIVIDED BY HANKEL FUNCTION.
C                        SOLUTION MULTIPLIED BY HANKEL FUNCTION BEFORE
C                        COMPUTING PROPAGATION LOSS.
C      *** ILYR  - INDEX FOR ARRAYS BETA, ZLYR, AND RHO
C      *** IPZ   - EVERY IPZ'TH VALUE IN DEPTH IS PRINTED
C      *** ISF   - STARTING FIELD FLAG
C                  = 0 - PROGRAM GENERATES GAUSSIAN STARTING FIELD
C                        AT RANGE = 0.0. SEE SUBROUTINE SFIELD.
C                  = 1 - USER SUPPLIES STARTING FIELD. SEE SUBROUTINE
C                        UFIELD.
C      *** ISFLD - STARTING FIELD PRINT FLAG
C                  = 0 - DO NOT PRINT STARTING FIELD
C                  = 1 - PRINT STARTING FIELD
```

```
C      *** ISVP  - SVP PRINT FLAG
C                = 0 - DO NOT PRINT SOUND VELOCITY PROFILE
C                = 1 - PRINT SOUND VELOCITY PROFILE
C      *** ITEMP - TEMPORARY VARIABLE
C      *** ITRK  - INDEX FOR ARRAY TRACK
C      *** ITYPEB- TYPE OF BOTTOM
C              0 - RIGID BOTTOM - PROGRAM SUPPLIES BOTTOM CONDITION
C              1 - NOT RIGID    - USER SUPPLIES BOTTOM CONDITION
C                               - SEE SUBROUTINE BCON
C              2 - NOT RIGID    - ABSORBING LAYER INTRODUCED
C                               - FOLLOWS CONTOUR OF BOTTOM
C              3 - NOT RIGID    - ABSORBING LAYER INTRODUCED - BUT
C                               - BOTTOM OF ABSORBING LAYER KEPT FLAT
C      *** ITYPES- TYPE OF SURFACE
C                = 0 - PRESSURE RELEASE. SCON SETS SURY AND SURX = 0.0
C                = .NE. 0 - USER INSERTS CODE IN SCON TO COMPUTE SURY AND SURX
C      *** IWZ   - INDEX INCREMENT OF RECEIVER SOLUTIONS TO BE WRITTEN ON DISK
C      *** IXSVP - ARRAY OF POINTERS WHICH POINT TO ENTRIES IN CSVP AND ZSVP
C                - IXSVP(1) POINTS TO BOTTOM DEPTH AND SPEED IN LAYER 1
C                - IXSVP(2) POINTS TO BOTTOM DEPTH AND SPEED IN LAYER 2
C                - ETC.
C      *** KSVP  - SVP PROFILE FLAG
C              0 - PROFILE IS ON CARDS - SEE SUBROUTINE SVP
C              1 - USER SUPPLIES PROFILE 1 - SEE SUBROUTINE USVP
C              2 - USER SUPPLIES PROFILE 2 - SEE SUBROUTINE USVP
C              .          .        .        .        .        .
C              N - USER SUPPLIES PROFILE N - SEE SUBROUTINE USVP
C      *** MLYR  - TEMPORARY - NUMBER OF LAYERS INVOLVED IN SPECIFIC CALCULATION
C      *** MM    - INDEX - MM+1 POINTS TO FIRST VALUE OF ARTIFICIAL ABSORBING
C                  LAYER IN ARRAY U
C      *** MXLYR - PARAMETER - MAXIMUM NUMBER OF LAYERS
C                            - MAX DIMENSION OF ARRAYS BETA,RHO,ZLYR,IXSVP
C      *** MXN   - PARAMETER - MAXIMUM DIMENSION OF C, X, Y, R12 AND U ARRAYS
C      *** MXSVP - PARAMETER - MAXIMUM DIMENSION OF ARRAYS CSVP AND ZSVP
C      *** MXTRK - PARAMETER - MAXIMUM DIMENSION OF ARRAY TRACK
C      *** N     - NUMBER OF EQUI-SPACED POINTS IN U
C                - INCLUDES BOTTOM POINT - DOES NOT INCLUDE SURFACE POINT
C      *** N1    - DIAGONAL ELEMENTS N1 THRU N WILL BE COMPUTED
C      *** NA    - NUMBER OF POINTS IN ABSORBING LAYER
C      *** NIU   - PARAMETER - INPUT UNIT NUMBER
C      *** NLYR  - NUMBER OF LAYERS
C      *** NOLD  - NUMBER OF RECEIVER DEPTHS ON ENTRY TO ROUTINE CRNK
C      *** NOU   - PARAMETER - OUTPUT UNIT NUMBER
C      *** NPU   - PARAMETER - PRINTER UNIT NUMBER
C      *** NSVP  - NUMBER OF POINTS IN CSVP AND ZSVP
C      *** NWSVP - NUMBER OF POINTS IN LAYER 1 SVP
C      *** NZ    - NUMBER OF SOLUTION DEPTHS TO BE WRITTEN ON DISK
C      *** OLDR  - RANGE INCREMENT AT START OF PROBLEM
C      *** PDR   - RANGE INCREMENT AT WHICH SOLUTION IS PRINTED - METERS
C      *** PDZ   - DEPTH INCREMENT AT WHICH SOLUTION IS PRINTED - METERS
C      *** PI    - THE VALUE OF PI
C      *** PL    - PROPAGATION LOSS - DB
C      *** R1    -       RANGE AT WHICH BOTTOM DEPTH IS AVAILABLE - METERS
C      *** R2    - NEXT RANGE AT WHICH BOTTOM DEPTH IS AVAILABLE - METERS
```

```
C      *** R12   - ARRAY OF DENSITY RATIOS
C      *** RA    - HORIZONTAL RANGE OF STARTING FIELD FROM SOURCE - METERS
C                - RA IS SET TO 0.0 IF STARTING FIELD IS GAUSSIAN. RA IS
C                - INCREMENTED AS SOLUTION IS MARCHED OUT IN RANGE.
C      *** RA+DR - RANGE TO WHICH SOLUTION IS TO BE ADVANCED - METERS
C      *** RHO   - ARRAY - DENSITY IN LAYER
C      *** RMAX  - MAXIMUM RANGE OF SOLUTION - METERS
C      *** RSVP  - NEXT RANGE AT WHICH SVP IS AVAILABLE - METERS
C      *** SURX  - COMPLEX PRESSURE AT SURFACE AT ADVANCED RANGE RA+DR
C      *** SURY  - COMPLEX PRESSURE AT SURFACE AT PRESENT RANGE RA
C      *** TEMP  - TEMPORARY VARIABLE - COMPLEX
C      *** THETA - SLOPE OF BOTTOM - RADIANS
C                  .EQ.0 -- FLAT BOTTOM
C                  .GT.0 -- SHALLOW TO DEEP
C                  .LT.0 -- DEEP TO SHALLOW
C      *** TM    - ARRAY - TIME OF DAY
C      *** TRACK - 2 DIM. ARRAY - RANGE AND DEPTH OF WATER - METERS
C      *** U     - ARRAY - COMPLEX ACOUSTIC PRESSURE FIELD
C      *** WDR   - RANGE STEP AT WHICH SOLUTION IS WRITTEN ON DISK - METERS
C      *** WDZ   - DEPTH INCREMENT AT WHICH SOLUTION IS WRITTEN ON DISK - METERS
C                  WDZ SHOULD BE SELECTED SO THAT PLOT PROGRAM DOES NOT
C                  INTERPOLATE BETWEEN WIDELY SPACED RECEIVERS.
C      *** X     - ARRAY - MAIN DIAGONAL OF MATRIX AT ADVANCED RANGE
C      *** XKO   - REFERENCE WAVE NUMBER
C      *** XPR   - RANGE AT WHICH SOLUTION IS PRINTED - METERS
C      *** XWR   - RANGE AT WHICH SOLUTION IS WRITTEN ON DISK - METERS
C      *** Y     - ARRAY - MAIN DIAGONAL OF MATRIX AT PRESENT RANGE
C      *** Z1    - DEPTH OF WATER AT RANGE R1 - METERS
C      *** Z2    - DEPTH OF WATER AT RANGE R2 - METERS
C      *** ZA    - DEPTH OF FIELD AT RANGE RA - METERS
C                - INITIAL DEPTH OF STARTING FIELD AT RANGE RA IS
C                  AS FOLLOWS:
C                - IF ITYPEB = 0 OR 1, ZA IS MAXIMUM DEPTH OF
C                  BOTTOM-MOST SEDIMENT LAYER AT INITIAL RANGE OF
C                  STARTING FIELD. IF ITYPEB = 2 OR 3, ZA IS MAXIMUM
C                  DEPTH OF ARTIFICIAL ABSORBING LAYER AT INITIAL
C                  RANGE OF STARTING FIELD. PROGRAM INSERTS LAYER.
C                  RHO AND BETA ARE OBTAINED FROM LAYER ABOVE.
C                  SPEED IS BOTTOM-MOST SPEED FROM LAYER ABOVE.
C                  AS SOLUTION PROGRESSES,
C                  ZA IS UPDATED IF OCEAN BOTTOM NOT FLAT. IF ITYPEB = 3,
C                  BOTTOM OF ABSORBING LAYER REMAINS FLAT.
C      *** ZLYR  - ARRAY - DEPTH TO BOTTOM OF LAYER - METERS
C      *** ZI    - DEPTH OF RECEIVER 'I' - METERS
C      *** ZS    - SOURCE DEPTH - METERS
C      *** ZSVP  - ARRAY - DEPTH OF SOUND VELOCITY - METERS
```

```
C       ****************************************************************
C       *** INPUT
C       ****************************************************************
C           INPUT UNIT NUMBER = NIU
C           INPUT FILE NAME   = IFD.IN
C           CONTENTS:  CARD IMAGES IN FREE FORMAT
C           CARD 1   :  FRQ,ZS,CO,ISF,RA,ZA,N,IHNK,ITYPEB,ITYPES
C           CARD 2   :  RMAX,DR,WDR,WDZ,PDR,PDZ,ISFLD,ISVP,IBOT
C           CARD 3   :  R1,Z1   **
C           CARD 4   :  R2,Z2     *    BOTTOM PROFILE
C             .           .       **   RANGE, WATER DEPTH (METERS)
C             .           .       *
C           CARD N   :    .     **
C           CARD N+1:   -1,-1
C           CARD N+2:   RSVP                         *****************
C           CARD N+3:   KSVP                                         *
C           CARD N+4:   NLYR                                         *
C           CARD N+5:   ZLYR(I),RHO(I),BETA(I)   **                  *
C           CARD N+6:   ZSVP(1),CSVP(1)            *                 **  REPEAT
C           CARD N+7:   ZSVP(2),CSVP(2)            **  REPEAT        *** FOR EACH
C             .             .       .              *** FOR EACH      **  PROFILE
C             .             .       .              **  LAYER         *
C             .             .       .              *                 *
C           CARD N+M:   ZSVP(J),CSVP(J)          **                  *
C                                                *****************
C
C       ****************************************************************
C       *** QUICK REFERENCE AND NOTES FOR CARD INPUT
C       *** UNITS: METERS AND METERS/SEC EXCEPT AS NOTED
C       ****************************************************************
C           FRQ = FREQUENCY (HZ)
C           ZS  = SOURCE DEPTH
C           CO  = REFERENCE SOUND SPEED. IF CO = 0.0, CO IS SET TO AVERAGE
C                 SPEED IN FIRST LAYER.
C           ISF = STARTING FIELD FLAG. 0 = GAUSSIAN. 1 = USER FIELD.
C                 IF ISF = 0, RA IS SET TO ZERO.
C           RA  = HORIZONTAL RANGE FROM SOURCE TO STARTING FIELD.
C                 RA IS SET TO 0.0 IF ISF = 0.
C           ZA  = DEPTH OF STARTING FIELD AT RANGE RA. IF ZA = 0.0, ZA IS SET
C                 TO MAX DEPTH OF BOTTOM LAYER IN FIRST PROFILE. IF ITYPEB =
C                 2 OR 3 AND ZA = 0.0, ZA IS SET TO (4/3)*MAX DEPTH OF BOTTOM
C                 LAYER. IF ITYPEB = 2 OR 3 AND ZA NOT ZERO, THE ARTIFICIAL
C                 BOTTOM LAYER IS EXTENDED TO ZA METERS PROVIDED THAT ZA
C                 IS GREATER THAN OR EQUAL TO MAX DEPTH OF BOTTOM LAYER
C                 IN FIRST PROFILE.
C           N   = NUMBER OF EQUISPACED RECEIVERS IN STARTING FIELD. IF N = 0,
C                 N IS SET SO THAT THE RECEIVER DEPTH INCREMENT IS LESS THAN
C                 OR EQUAL TO 1/4 WAVELENGTH. IF N IS GREATER THAN MXN,
C                 N IS SET TO MXN.
C          IHNK = HANKEL FUNCTION FLAG. IHNK = 0, DON'T USE HANKEL FUNCTION.
C                 IHNK = 1, DIVIDE STARTING FIELD BY HANKEL FUNCTION, THEN
C                 MULTIPLY THE SOLUTION FIELD BY HANKEL FUNCTION BEFORE
C                 COMPUTING PROPAGATION LOSS. IF STARTING FIELD IS GAUSSIAN,
C                 IHNK SHOULD BE SET TO 0. IF STARTING FIELD IS ELLIPTIC,
```

```
C            IHNK SHOULD BE SET TO 1.
C      ITYPEB = TYPE OF BOTTOM PROCESSING.
C             = 0 - RIGID BOTTOM. PROGRAM SUPPLIES BOTTOM CONDITION.
C             = 1 - USER SUPPLIES BOTTOM CONDITION. USER WRITES SUBROUTINE
C                   BCON.
C             = 2 - ARTIFICIAL ABSORBING BOTTOM INTRODUCED. FOLLOWS CONTOUR
C                   OF WATER/BOTTOM INTERFACE.
C             = 3 - ARTIFICIAL ABSORBING BOTTOM INTRODUCED. BOTTOM OF
C                   LAYER KEPT FLAT.
C      ITYPES = TYPE OF SURFACE
C             =   0 - PRESSURE RELEASE. SCON SETS SURY AND SURX = 0.0
C             NOT 0 - USER INSERTS CODE IN SCON TO COMPUTE SURY AND SURX
C
C
C        RMAX = MAXIMUM RANGE OF SOLUTION
C        DR   = RANGE STEP. IF DR = 0, DR IS SET TO 1/2 WAVELENGTH.
C               IF BOTTOM OF PROBLEM IS NOT FLAT, DR IS RECOMPUTED
C               SO THAT MAX DEPTH IS EITHER INCREMENTED OR DECREMENTED
C               BY DZ. SOLUTION IS COMPUTED EVERY DR METERS.
C        WDR  = RANGE STEP AT WHICH SOLUTION IS WRITTEN ON DISK. IF WDR NOT 0,
C               AN OUTPUT DISK FILE IS ASSIGNED.
C               ROUNDED TO NEAREST DR.
C        WDZ  = DEPTH INCREMENT AT WHICH SOLUTION IS WRITTEN ON DISK.
C               ROUNDED TO NEAREST DZ.
C        PDR  = RANGE STEP AT WHICH SOLUTION IS PRINTED.
C               ROUNDED TO NEAREST DR.
C        PDZ  = DEPTH INCREMENT AT WHICH SOLUTION IS PRINTED.
C               ROUNDED TO NEAREST DZ.
C       ISFLD = 0 - DON'T PRINT STARTING FIELD.
C             = 1 - PRINT STARTING FIELD.
C       ISVP  = 0 - DON'T PRINT SOUND VELOCITY PROFILE.
C             = 1 - PRINT SOUND VELOCITY PROFILE.
C        IBOT = 0 - DON'T PRINT BOTTOM DEPTHS.
C             = 1 - PRINT BOTTOM DEPTHS.
C
C
C        R1,Z1 = BOTTOM PROFILE. FIRST RANGE AND DEPTH OF WATER.
C        R2,Z2 = ETC. (-1,-1) MARKS THE END OF THE BOTTOM PROFILE.
C        RSVP  = RANGE OF SVP
C        KSVP  = SVP FLAG.
C              = 0 - SVP IN INPUT RUNSTREAM
C              = NOT ZERO - PROFILE (CARDS N+4 THRU N+M) IS SUPPLIED BY
C                USER. USER WRITES SUBROUTINE USVP. KSVP MAY BE USED IN
C                COMPUTED GOTO STATEMENT TO TRANSFER CONTROL IN USVP.
C        NLYR  = NUMBER OF LAYERS. IF ITYPEB = 2 OR 3, PROGRAM INSERTS
C                AN ARTIFICIAL LAYER AND INCREMENTS NLYR BY 1.
C      ZLYR(I) = MAX DEPTH OF LAYER I IN PROFILE
C      RHO(I)  = DENSITY IN LAYER I (G/CM**3)
C      BETA(I) = ATTENUATION IN LAYER I (DB/WAVELENGTH)
C                IF BETA(I) IS NEGATIVE, ATTENUATION IS COMPUTED.
C      ZSVP    = DEPTH ARRAY FOR SOUND SPEED
C      CSVP    = SPEED ARRAY FOR SOUND SPEED
C      ZSVP(1) = DEPTH OF WATER TO TOP OF LAYER I
C      CSVP(1) = SPEED OF SOUND AT TOP OF LAYER I
```

```
C       ZSVP(J) = DEPTH OF WATER TO BOTTOM OF LAYER I
C       CSVP(J) = SPEED OF SOUND AT BOTTOM OF LAYER I
C                 IF ONLY ONE SVP INPUTTED, IT IS USED THRU ENTIRE PROBLEM.
C                 IF MORE THAN ONE SVP INPUTTED, LAST SVP IS USED THRU REMAINDER
C                 OF PROBLEM.
C
C      *******************************************************************
C      *** OUTPUT
C      *******************************************************************
C          OUTPUT UNIT NUMBER = NOU
C          OUTPUT FILE NAME   = IFD.OUT
C          CONTENTS:  AS FOLLOWS - UNFORMATTED
C
C            FRQ,ZS,CO,ISF,RA,ZA,N,IHNK,ITYPEB,ITYPES,RMAX,DR,WDR,DZ,NLYR,ZLYR,
C            RHO,BETA
C            NZ,RA,WDZ,(U(I),I=IWZ,N,IWZ) - (FOR EACH WRITE RANGE WDR)
C
C          PRINTER UNIT NUMBER = NPU
C      *******************************************************************
C
       PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C           NOU=2,NPU=6
       COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C          U,X,Y
       COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C        BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C        ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C        RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C        X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
C
       DATA PI/3.141592654/,DEG/57.29578/
       BYTE TM(8)
C
C      *** TIME OF DAY AT START OF RUN
       CALL TIME(TM)
10     FORMAT(5X,8A1/)
C
C      *** READ INPUT PARAMETERS
       CALL ASSIGN(NIU,'IFD.IN')
       READ(NIU,*) FRQ,ZS,CO,ISF,RA,ZA,N,IHNK,ITYPEB,ITYPES
       READ(NIU,*) RMAX,DR,WDR,WDZ,PDR,PDZ,ISFLD,ISVP,IBOT
C
C      *** IF GAUSSIAN STARTING FIELD, RA MUST BE 0.
       IF(ISF.EQ.0) RA=0.0
C
C      *** READ BOTTOM PROFILE - RANGE,DEPTH
       DO 22 I=1,MXTRK
       READ(NIU,*) TRACK(I,1),TRACK(I,2)
       ITRK=I
C      *** END OF PROFILE?
       IF(TRACK(I,1).LT.0.0) GO TO 23
C      *** NO
22     CONTINUE
C      *** ERROR?
```

```
23    IF(ITRK.EQ.1.OR.ITRK.GT.MXTRK) GO TO 28
C     *** NO. EXTEND LAST DEPTH BEYOND MAX RANGE.
      TRACK(ITRK,1)=1.0E+38
      TRACK(ITRK,2)=TRACK(ITRK-1,2)
      ITRK=1
      R2=TRACK(ITRK,1)
      Z2=TRACK(ITRK,2)
C     *** FIND BOTTOM SEGMENT WHICH CONTAINS STARTING RANGE
25    R1=R2
      Z1=Z2
      ITRK=ITRK+1
      IF(ITRK.GT.MXTRK) GO TO 28
      R2=TRACK(ITRK,1)
      Z2=TRACK(ITRK,2)
C     *** ADVANCE TRACK IF NECESSARY SO THAT R1.LE.RA.LT.R2
      IF(RA.GE.R2) GO TO 25
C     *** R1 MUST BE LE RA
      IF(R1.LE.RA) GO TO 30
      WRITE(NPU,27)
27    FORMAT(1X,'DEPTH OF BOTTOM AT INITIAL RANGE MISSING')
      STOP
28    CONTINUE
      ITEMP=MXTRK
      WRITE(NPU,29) ITEMP
29    FORMAT(1X,'ERROR. BOTTOM PROFILE MISSING OR TOO MANY POINTS.
     CMAX IS ',I5)
      STOP
30    CONTINUE
C
C     *** COMPUTE SLOPE OF BOTTOM
      THETA=ATAN2(Z2-Z1,R2-R1)
C
C     *** READ RANGE OF SVP
32    READ(NIU,*,END=33) RSVP
C     *** SVP BEYOND START RANGE?
      IF(RSVP.GT.RA) GO TO 33
C     *** NO. DETERMINE IF SVP IN RUNSTREAM OR SUPPLIED BY SUBROUTINE USVP.
      READ(NIU,*,END=33) KSVP
      IF(KSVP.EQ.0)CALL SVP(NLYR,ZLYR,RHO,BETA,IXSVP,NSVP,ZSVP,CSVP)
      IF(KSVP.NE.0)CALL USVP
C     *** ERROR IN SVP?
      IF(NSVP.NE.0) GO TO 35
C     *** YES.
33    WRITE(NPU,34) RA
34    FORMAT(1X,'SVP MISSING OR INPUT ERROR. RANGE = ',F8.1,' M.')
      STOP
C
35    CONTINUE
C     *** SAVE POINTER TO LAST SVP VALUE IN FIRST LAYER.
      NWSVP=IXSVP(1)
C
C     *** IF CO NOT SPECIFIED, SET CO TO AVERAGE SPEED OF FIRST PROFILE
C     *** IN FIRST LAYER AT INITIAL RANGE
      IF(CO.NE.0.0) GO TO 45
```

```
      DO 40 I=2,NWSVP
      CO=CO+(ZSVP(I)-ZSVP(I-1))*(CSVP(I-1)+.5*(CSVP(I)-CSVP(I-1)))
40    CONTINUE
      CO=CO/ZSVP(NWSVP)
45    CONTINUE
C
C     *** COMPUTE REFERENCE WAVE NUMBER
      XKO=2.0*PI*FRQ/CO
C
C     *** COMPUTE ATTENUATION - SACLANT MEMO SM-121 (JENSEN + FERLA)
C     *** MODIFIED AS FOLLOWS:
C     *** IF INPUTTED BETA IS LT 0.0, ALPHA IS COMPUTED IN DB/METER
C         AND USED FOR BETA
      ALPHA=FRQ*FRQ*(.007+(.155*1.7)/(1.7*1.7+FRQ*FRQ*.000001))*1.0E-09
C
C     *** ADJUST LAYER DEPTHS IN CASE BOTTOM SLOPES AND RA.NE.RSVP
C     *** ASSUMES LAYERS ARE PARALLEL AND FOLLOW BOTTOM CONTOUR.
C     *** LAYER DEPTHS ARE ENTERED WITH SVP.
      DO 50 ILYR=1,NLYR
50    ZLYR(ILYR)=ZLYR(ILYR)+(RA-RSVP)*TAN(THETA)
C
C     *** GET RANGE OF NEXT SVP
      READ(NIU,*,END=55) RSVP
      GO TO 56
C     *** ONLY ONE SVP - SET RSVP LARGE SO SAME SVP USED FOR WHOLE PROBLEM
55    RSVP=1.0E+38
56    CONTINUE
C
C     *** IF STARTING FIELD IS BEYOND NEXT SVP, GO BACK AND GET NEXT SVP
      IF(RSVP.LE.RA) GO TO 32
C
C     *** IF ITYPEB = 2 OR 3, AND ZA = 0, EXTEND BOTTOM BY SETTING ZA TO
C     *** 4/3 MAX DEPTH
C     *** SEE NORDA TECH NOTE 12, JAN 78, H. K. BROCK
C     *** IF ITYPEB = 2 OR 3 AND ZA NOT 0, EXTEND BOTTOM TO ZA METERS
      IF(ITYPEB.LT.2) GO TO 60
      IF(ITYPEB.GT.3) GO TO 60
C     *** EXTEND BOTTOM TO ZA METERS IF ZA NOT ZERO
C     *** EXTEND BOTTOM 4/3 MAX DEPTH IF ZA = ZERO
      IF(ZA.EQ.0.0) ZA=4.0*ZLYR(NLYR)/3.0
      IF(ZA.GE.ZLYR(NLYR)) GO TO 58
C     *** USER ATTEMPTED TO EXTEND DEPTH IN NEGATIVE DIRECTION
      WRITE(NPU,57)
57    FORMAT(1X,'ERROR. ZA RESET TO MAX DEPTH OF BOTTOM LAYER.')
      ZA=ZLYR(NLYR)
C     *** INSERT PARAMETERS FOR EXTENDED BOTTOM IN APPROPRIATE ARRAYS
58    NLYR=NLYR+1
C     *** STORE DEPTH OF ARTIFICIAL ABSORBING LAYER
      ZLYR(NLYR)=ZA
C     *** USE SAME DENSITY AND ATTENUATION AS LAYER ABOVE
      RHO(NLYR)=RHO(NLYR-1)
      BETA(NLYR)=BETA(NLYR-1)
C     *** USE BOTTOM SPEED OF LAYER ABOVE
      NSVP=NSVP+1
```

```
      IXSVP(NLYR)=NSVP
      ZSVP(NSVP)=ZLYR(NLYR)
      CSVP(NSVP)=CSVP(NSVP-1)
      GO TO 62
60    CONTINUE
C     *** IF BOTTOM NOT EXTENDED AND ZA=0, SET ZA TO MAX DEPTH INPUTTED
C     *** WITH FIRST SVP
      IF(ZA.EQ.0.0) ZA=ZLYR(NLYR)
C
C     *** IF N NOT SPECIFIED, SET N SO THAT DZ .LE. 1/4 WAVELENGTH
62    IF(N.EQ.0) N=(4*ZA*FRQ/C0)+1
      IF(N.LE.MXN) GO TO 65
      ITEMP=MXN
      WRITE(NPU,64) ITEMP
64    FORMAT(' ERROR. N TOO LARGE. N RESET TO ',I4'.')
      N=MXN
65    CONTINUE
C
C     *** COMPUTE RECEIVER DEPTH INCREMENT - DZ MAY BE SUCH THAT RECEIVERS DO
C     *** NOT LIE EXACTLY ON LAYER INTERFACES
      DZ=ZA/N
C
C     *** IF BOTTOM IS FLAT, RANGE STEP MAY BE SPECIFIED
C     *** IF RANGE STEP NOT SPECIFIED, SET IT EQUAL TO 1/2 WAVELENGTH   !???
      IF(DR.EQ.0.0) DR=.5*C0/FRQ
C     *** IF BOTTOM IS NOT FLAT, COMPUTE RANGE STEP
C     *** IT MAY BE NECESSARY TO REDUCE DZ SUCH THAT DR IS SMALL ENOUGH
C     *** TO PRODUCE ACCURATE RESULTS
      IF(ITYPEB.NE.3.AND.THETA.NE.0.0) DR=ABS(DZ/TAN(THETA))
C     *** COMPUTE DEPTH INCREMENT FOR ADJUSTING LAYERS
      DZZ=DR*TAN(THETA)
C
C     *** GET STARTING FIELD
      IF(ISF.EQ.0) CALL SFIELD(FRQ,C0,ZS,N,DZ,U)
      IF(ISF.NE.0) CALL UFIELD
      IF(IHNK.EQ.0) GO TO 69
C
C     *** DIVIDE STARTING FIELD BY HANKEL FUNCTION IF REQUESTED BY USER
      HNK=HNKL(XK0*RA)
      DO 68 I=1,N
      U(I)=U(I)/HNK
68    CONTINUE
69    CONTINUE
C
C     *** COMPUTE DEPTH WRITE INCREMENT TO NEAREST DZ
      IF(WDZ.LT.DZ)WDZ=DZ
      IWZ=WDZ/DZ+.5
      WDZ=IWZ*DZ
C
C     *** COMPUTE DEPTH PRINT INCREMENT TO NEAREST DZ
      IPZ=PDZ/DZ+.5
      IF(PDZ.GT.0.0.AND.IPZ.EQ.0) IPZ=1
      PDZ=IPZ*DZ
C
```

```
C      *** PRINT PROBLEM PARAMETERS
       WRITE(NPU,72)
72     FORMAT(//1X,'IFD SOLUTION')
       IF(ISF.EQ.0) WRITE(NPU,73)
73     FORMAT(1X,'GAUSSIAN STARTING FIELD')
       IF(ISF.NE.0) WRITE(NPU,74)
74     FORMAT(1X,'USER STARTING FIELD')
       IF(IHNK.NE.0) WRITE(NPU,75)
75     FORMAT(1X,'STARTING FIELD DIVIDED BY HANKEL FUNCTION')
       IF(ITYPES.NE.0) WRITE(NPU,76)
76     FORMAT(1X,'USER SURFACE CONDITION')
       IF(ITYPEB.EQ.1) WRITE(NPU,77)
77     FORMAT(1X,'USER BOTTOM CONDITION')
       WRITE(NPU,80) FRQ,ZS,CO,RA,ZA,N
80     FORMAT(1X,'FRQ     = ',F7.1' HZ',/,1X,
      C'ZS      = ',F7.1' M',/,1X,
      C'CO      = ',F7.1,' M/SEC',/,1X,
      C'RA      = ',F7.1,' M',/,1X,
      C'ZA      = ',F7.1,' M',/,1X,
      C'N       = ',I5)
       WRITE(NPU,81) DR,WDR,PDR,DZ,WDZ,PDZ,ITYPEB,ITYPES,RMAX
81     FORMAT(1X,'DR      = ',F7.1,' M',/,1X,
      C'WDR     = ',F7.1,' M',/,1X,
      C'PDR     = ',F7.1,' M',/,1X,
      C'DZ      = ',F7.1,' M',/,1X,
      C'WDZ     = ',F7.1,' M',/,1X,
      C'PDZ     = ',F7.1,' M',/,1X,
      C'ITYPEB =      ',I1,/,1X,
      C'ITYPES =      ',I1,/,1X,
      C'RMAX   =',F8.1,' M',/,/,
      C2X,'LAYER  MAX DEPTH(M)       DENSITY(G/CM**3)  ATT(DB/WL)',/)
       DO 85 ILYR   =1,NLYR
       WRITE(NPU,84)ILYR,ZLYR(ILYR),RHO(ILYR),BETA(ILYR)
84     FORMAT(2X,I3,3X,E13.6,5X,E13.6,5X,E13.6)
85     CONTINUE
C
C      *** PRINT BOTTOM DEPTHS IF REQUESTED
       IF(IBOT.EQ.0) GO TO 86
       TH=THETA*DEG
       WRITE(NPU,123) R1,Z1,R2,Z2,TH
86     IF(ISFLD.EQ.0) GO TO 92
C
C      *** PRINT STARTING FIELD
       WRITE(NPU,87)
87     FORMAT(/,1X,'STARTING FIELD')
       DO 90 I=1,N
       ZI=I*DZ
       WRITE(NPU,89) I,ZI,U(I)
89     FORMAT(1X,I4,3X,F10.2,3X,'(',E12.5,2X,E12.5,' )')
90     CONTINUE
92     CONTINUE
C
       IF(ISVP.EQ.0) GO TO 96
C      *** PRINT SVP
```

```
      WRITE(NPU,93) RA
93    FORMAT(//1X,'SOUND VELOCITY PROFILE AT RANGE ',F8.1,' METERS',/)
      DO 95 I=1,NSVP
      WRITE(NPU,94) I,ZSVP(I),CSVP(I)
94    FORMAT(1X,I4,3X,F8.1,3X,F8.1)
95    CONTINUE
96    CONTINUE
C
C     *** COMPUTE 'B' COEFFICIENT
      BCOF=CMPLX(0.0,.5/XKO)
C
C     *** ASSIGN OUTPUT FILE IF OUTPUT REQUESTED
      IF(WDR.EQ.0.0)GO TO 98
      CALL ASSIGN(NOU,'IFD.OUT')
C
C     *** WRITE SELECTED PARAMETERS FOLLOWED BY STARTING FIELD
      WRITE(NOU)FRQ,ZS,CO,ISF,RA,ZA,N,IHNK,ITYPEB,ITYPES,RMAX,DR,WDR,DZ,
     CNLYR,ZLYR,RHO,BETA
      NZ=N/IWZ
      WRITE(NOU) NZ,RA,WDZ,(U(I),I=IWZ,N,IWZ)
C
98    CONTINUE
C     *** INITIALIZE RANGE VARIABLE AT WHICH SOLUTION IS TO BE PRINTED
      XPR=RA+PDR
C
C     *** INITIALIZE RANGE VARIABLE AT WHICH SOLUTION IS TO BE RECORDED
      XWR=RA+WDR
      IF(WDR.EQ.0.0) XWR=RA+RMAX+1.0
C
C     *** SAVE RANGE STEP
      OLDR=DR
C
C     *** INITIALIZE PARAMS FOR DIAG THEN COMPUTE MAIN DIAGONALS X AND Y
      N1=1
      DR1=DR
C     *** COMPUTE X DIAGONAL
      CALL DIAG
C     *** COMPUTE Y DIAGONAL
      CALL DIAG
C
C     *** MAIN LOOP STARTS HERE ***
C     *** SOLUTION WILL BE ADVANCED FROM RANGE RA TO RANGE RA+DR
C
100   CONTINUE
      IDIAG=0
C
C     *** DOES SVP CHANGE BEFORE BOTTOM PROFILE?
      IF(RSVP.GE.R2) GO TO 105
C     *** YES. DOES SVP CHANGE BEFORE NEXT SOLUTION RANGE?
      IF(RA+DR.LE.RSVP) GO TO 131
C     *** YES. ADJUST DR SO THAT SOLUTION ADVANCES TO RSVP
      DR=RSVP-RA
      GO TO 126
C     *** BOTTOM PROFILE CHANGES BEFORE OR AT SAME RANGE AS SVP
```

```
105   CONTINUE
C
C     *** DETERMINE IF BOTTOM DEPTHS ARE TO BE UPDATED
C     *** FIRST PASS - RA WILL BE .LT. R2
C     *** UPDATE BOTTOM DEPTHS?
      IF(RA.GE.R2) GO TO 110
C
C     *** NO.
C     *** DETERMINE IF RANGE STEP TOO LARGE
      IF(RA+DR.LE.R2) GO TO 131
C     *** RANGE STEP IS TOO LARGE - RESET DR - ADV SOLUTION TO R2
      DR=R2-RA  ! ...........
      GO TO 126
C
C     *** UPDATE BOTTOM DEPTHS
110   CONTINUE
      R1=R2
      Z1=Z2
      ITRK=ITRK+1
      R2=TRACK(ITRK,1)
      Z2=TRACK(ITRK,2)
C     *** TWO DEPTHS AT SAME RANGE INDICATE VERTICAL DISCONTINUITY.
C     *** ADVANCE TRACK FORWARD.
      IF(R1.EQ.R2) GO TO 110
C
C     *** RESTORE DR
      DR=OLDR
C
C     *** COMPUTE SLOPE OF BOTTOM
      THETA=ATAN2(Z2-Z1,R2-R1)
C
C     *** IF BOTTOM IS NOT FLAT, COMPUTE NEW RANGE STEP
      IF(THETA.EQ.0) GO TO 120
C
C     *** IF BOTTOM OF ARTIFICIAL LAYER IS FLAT, DO NOT RECOMPUTE DR.
      IF(ITYPEB.EQ.3) GO TO 120
      DR=ABS(DZ*COS(THETA)/SIN(THETA))
C
C     *** IF RANGE STEP TOO LARGE, PRINT WARNING MESSAGE. RECOMPUTE DR.
      IF(RA+DR.LE.R2) GO TO 120
      WRITE(NPU,115)
115   FORMAT(1X,'RANGE STEP TOO LARGE FOR BOTTOM IRREGULARITIES')
C     STOP  !...................................
      DR=R2-RA ! ..................
C
120   CONTINUE
C
C     *** PRINT BOTTOM DEPTHS
      IF(IBOT.EQ.0) GO TO 126
      TH=THETA*DEG
      WRITE(NPU,123) R1,Z1,R2,Z2,TH
123   FORMAT(//1X,'BOTTOM DEPTHS ',//,1X,
     C'R1     = ',F8.1,' M',/,1X,
     C'Z1     = ',F8.1,' M',/,1X,
```

```
     C'R2    = ',F8.1,' M',/,1X,
     C'Z2    = ',F8.1,' M',/,1X,
     C'THETA = ',F8.1,' DEG')
126   CONTINUE
      DZZ=DR*TAN(THETA)
      WRITE(NPU,128) DR,RA
128   FORMAT(1X,'RANGE STEP = ',F4.0,' M AT RANGE ',F8.1,' M')
C
C     *** ADVANCE RANGE ONE STEP
130   RA=RA+DR
      IDIAG=1
      GO TO 132
131   CONTINUE
      RA=RA+DR
132   CONTINUE
C     *** READ NEW SVP PROFILE FLAG?
      IF(RA.GE.RSVP) READ(NIU,*,END=33) KSVP
C
C     *** IF KSVP NOT 0, SUBROUTINE USVP IS CALLED FOR SVP. USER IS
C     *** RESPONSIBLE FOR ADJUSTING DEPTHS OF LAYERS WHEN BOTTOM
C     *** IS NOT FLAT.
      IF(KSVP.EQ.0) GO TO 134
      CALL USVP
      IDIAG=1
      GO TO 148
C
C     *** NEW SVP AT ADVANCED RANGE?
134   IF(RA.GE.RSVP) GO TO 145  !....................
C
C     *** NO. IS BOTTOM FLAT?
135   IF(THETA.EQ.0.0) GO TO 166
      MLYR=NLYR
      IF(ITYPEB.EQ.3) MLYR=NLYR-1
C
C     *** UPDATE DEPTHS OF LAYERS
C     *** ASSUMES LAYERS ARE PARALLEL AND FOLLOW BOTTOM CONTOUR
C     *** IF ITYPEB = 3, BOTTOM OF ARTIFICIAL LAYER REMAINS FLAT.
      DO 138 ILYR=1,MLYR
      ZLYR(ILYR)=ZLYR(ILYR)+DZZ
138   CONTINUE
C     N1=ZLYR(1)/DZ
      N1=1
      ZA=ZLYR(NLYR)
      IF(ITYPEB.EQ.3.AND.ZLYR(NLYR).LT.ZLYR(MLYR))WRITE(NPU,139)
139   FORMAT(1X,'ERR. DEPTH OF ARTIFICIAL LAYER LESS THAN LAYER ABOVE')
C
C     *** ADJUST DEPTHS OF PROFILES IN SLOPING LAYERS
C     *** ASSUMES SAME SVP IN LAYERS
      NWSVP=IXSVP(1)
      DO 140 I=NWSVP+1,NSVP
      ZSVP(I)=ZSVP(I)+DZZ
140   CONTINUE
      GO TO 167
C
```

```
C     *** GET NEXT SVP
145   CONTINUE
      CALL SVP(NLYR,ZLYR,RHO,BETA,IXSVP,NSVP,ZSVP,CSVP)
      IDIAG=1
148   CONTINUE
C     *** ERROR DETECTED?
      IF(NSVP.EQ.0) GO TO 33
      IF(ITYPEB.NE.3) ZA=ZA+DZZ
C     *** NO. READ RANGE OF NEXT PROFILE?
      IF(RA.GE.RSVP) READ(NIU,*,END=150) RSVP
149   CONTINUE
C     *** ARTIFICIAL ABSORBING LAYER?
      IF(ITYPEB.LT.2) GO TO 155
      IF(ITYPEB.GT.3) GO TO 155
C     *** YES. UPDATE DENSITY, ATTEN AND SPEED.
      IF(ZA.LT.ZLYR(NLYR)) GO TO 155
      NLYR=NLYR+1
      ZLYR(NLYR)=ZA
      RHO(NLYR)=RHO(NLYR-1)
      BETA(NLYR)=BETA(NLYR-1)
      NSVP=NSVP+1
      IXSVP(NLYR)=NSVP
      ZSVP(NSVP)=ZLYR(NLYR)
      CSVP(NSVP)=CSVP(NSVP-1)
      GO TO 155
C     *** SET RSVP LARGE SO THAT LAST PROFILE IS USED FOR REMAINDER OF PROBLEM
150   RSVP=1.0E+38
      GO TO 149
C
C     *** PRINT SVP
155   IF(ISVP.EQ.0) GO TO 165
      WRITE(NPU,93) RA
      DO 160 I=1,NSVP
      WRITE(NPU,94) I,ZSVP(I),CSVP(I)
160   CONTINUE
165   CONTINUE
C
C     *** IF NEW DR AND/OR SVP - UPDATE X AND Y DIAGONALS
166   IF(IDIAG.EQ.0) GO TO 170
      N1=1
167   CALL DIAG
      DR1=DR
170   CONTINUE
C
C     *** ADVANCE SOLUTION ONE STEP FORWARD
      NOLD=N
      CALL CRNK
C
C     *** APPLY ABSORPTION IF ITYPEB = 2 OR 3
      IF(ITYPEB.LT.2) GO TO 185
      IF(ITYPEB.GT.3) GO TO 185
      MM=ZLYR(NLYR-1)/DZ
      NA=N-MM
      IF(NA.GT.0) GO TO 175
```

```
      IF(NA.EQ.0) GO TO 185
      WRITE(NPU,174) RA
174   FORMAT(1X,'ERR IN THICKNESS OF ABSORBING LAYER AT ',F8.1,' M')
      STOP
175   CONTINUE
C     *** SEE AESD PE MODEL BY BROCK - NORDA TECH NOTE 12 - JAN 78
      DO 180 I=1,NA
      ATT=EXP(-.01*DR*EXP(-((I-NA)/(NA/3.0))**2.0))
      U(MM+I)=U(MM+I)*ATT
180   CONTINUE
185   CONTINUE
C
C     *** TERMINATE RUN IF N HAS BEEN DECREMENTED TO 5 - ARBITRARY
      IF(N.GT.5) GO TO 200
      WRITE(NPU,190)
190   FORMAT(1X,'PROGRAM TERMINATED - ONLY FIVE POINTS REMAIN')
      GO TO 400
200   CONTINUE
C
C     *** TERMINATE RUN IF N HAS BEEN INCREMENTED BEYOND MAXIMUM
      IF(N.LE.MXN) GO TO 250
      WRITE(NPU,210)RA
210   FORMAT(1X,'MAX N EXCEEDED AT RANGE ',F10.2,' M.')
      GO TO 400
250   CONTINUE
C
C     *** DETERMINE IF NEW POINT ADDED
      IF(N.LE.NOLD) GO TO 255
      N1=NOLD
      CALL DIAG
      DR1=DR
255   CONTINUE
C
C     *** IF SOLUTION IS TO BE WRITTEN ON DISK,
C     *** WRITE SELECTED PRESSURE FIELD AT RANGE RA
      IF(XWR.GT.RA) GO TO 260
      WRITE(NOU) NZ,RA,WDZ,(U(I),I=IWZ,N,IWZ)
C
C     *** DETERMINE NEXT RANGE AT WHICH TO WRITE SOLUTION ON DISK
      XWR=XWR+WDR
260   CONTINUE
C
C     *** DETERMINE IF SOLUTION IS TO BE PRINTED
      IF(XPR.GT.RA.OR.IPZ.EQ.0.OR.PDR.EQ.0.0) GO TO 350
C
C     *** PRINT RANGE
      RTEMP=RA/1000.0
      WRITE(NPU,270) RTEMP
270   FORMAT(/5X,'RANGE =',E15.8,' KM. '/)
C
C     *** COMPUTE HANKEL FUNCTION
      IF(IHNK.NE.0) HNK=HNKL(XKO*RA)
C
C     *** COMPUTE AND PRINT PROPAGATION LOSS AT EACH IPZ'TH DEPTH
```

```
      WRITE(NPU,275)
275   FORMAT(6X,'I',9X,'Z(I)',6X,'LOSS(DB)',14X,'U(I)')
      DO 300 I=IPZ,N,IPZ
      ZI=I*DZ
      PL=CABS(U(I))
      IF(IHNK.EQ.1) PL=CABS(U(I)*HNK)
      IF(PL.LE.0.0) GO TO 288
      PL=-20.0*ALOG10(PL)
      IF(IHNK.EQ.0) PL=PL+10.0*ALOG10(RA)
      GO TO 289
288   PL=999.9
289   WRITE(NPU,295) I,ZI,PL,U(I)
295   FORMAT(2X,I5,(3X,F10.2,3X,F10.3),3X,'(',E12.5,2X,E12.5,' )')
300   CONTINUE
C
C     *** DETERMINE NEXT RANGE AT WHICH TO PRINT SOLUTION
      XPR=XPR+PDR
C
C     *** TERMINATE RUN IF SOLUTION AT MAXIMUM RANGE HAS BEEN OBTAINED
350   IF(RA.LT.RMAX) GO TO 100
C
C     *** TERMINATE RUN
400   IF(WDR.NE.0)CALL CLOSE(NOU)
      WRITE(NPU,10) TM
      CALL TIME(TM)
      WRITE(NPU,10) TM
      WRITE(NPU,401)
401   FORMAT(1X,'END OF RUN')
      STOP
      END
```

```
      SUBROUTINE DIAG
C     ******************************************************************
C     ******************************************************************
C     * COMPUTES RANGE AND DEPTH DEPENDENT MAIN DIAGONALS OF MATRICES   *
C     * DIAG IS CALLED WHENEVER BOTTOM DEPTH OR SVP CHANGES             *
C     ******************************************************************
C     ******************************************************************
C
C     *** SEE MAIN PROGRAM FOR DEFINITIONS
C     *** SUBROUTINE DIAG RETURNS:
C         ACOFX - COEFFICIENT 'A' AT BOTTOM - AT RANGE RA
C         ACOFY - COEFFICIENT 'A' AT BOTTOM - AT RANGE RA-DR
C         BTA   - ARRAY - PARTIAL SOLUTION OF SYSTEM OF EQUATIONS
C         X     - ARRAY - MAIN DIAGONAL OF MATRIX AT RANGE RA
C         Y     - ARRAY - MAIN DIAGONAL OF MATRIX AT RANGE RA-DR
C     ******************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),C0,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XK0,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
      COMPLEX B,XN1,XN2,EYE
      DATA IEX/0/
C
C     *** COMPUTE NEW X AND Y DIAGONALS
C
      EYE=CMPLX(0.0,1.0)
C     *** GET INDICES, DENSITY, AND ATTENUATION FOR FIRST LAYER
      ILYR=1
      L=1
      M=IXSVP(1)
      R1=RHO(1)
      BETA1=BETA(1)
      DO 10 I=N1,N
C     *** TRANSFORM X INTO Y
      Y(I)=-X(I)-EYE*2.0*DZ*DZ*XK0*(1.0+R12(I))*(1.0/DR1+1.0/DR)
10    CONTINUE
      DO 100 I=N1,N
C     *** ZI IS RECEIVER DEPTH
      ZI=I*DZ
C     *** IS RECEIVER IN THIS LAYER?
20    IF(ZI.LE.ZLYR(ILYR)) GO TO 49
C     *** NO. ALL LAYERS CHECKED?
      IF(ILYR.EQ.NLYR) GO TO 52
C     *** NO. SET INDICES, DENSITY, AND ATTENUATION FOR NEXT LAYER.
      ILYR=ILYR+1
      L=M+1
      M=IXSVP(ILYR)
      R1=RHO(ILYR)
      BETA1=BETA(ILYR)
```

```
      GO TO 20
C     *** DEPTH ZI IS IN THIS LAYER.
49    CONTINUE
C     *** DETERMINE SOUND SPEED CI AT DEPTH ZI
      DO 50 J=L,M
      IF(ZI.GT.ZSVP(J)) GO TO 50
      L=J
C     *** INTERPOLATE
      CI=CSVP(J-1)+(CSVP(J)-CSVP(J-1))*(ZI-ZSVP(J-1))/(ZSVP(J)-ZSVP(J-1)
     C)
      GO TO 60
50    CONTINUE
C     *** EXTRAPOLATE
52    CONTINUE
      IF(ZSVP(M).NE.ZSVP(M-1))GO TO 53
      CI=CSVP(M)
      GO TO 54
53    CI=CSVP(M-1)+(CSVP(M)-CSVP(M-1))*(ZI-ZSVP(M-1))/
     C(ZSVP(M)-ZSVP(M-1))
54    IF(IEX.EQ.1) GO TO 60
      WRITE(NPU,56)
56    FORMAT(1X,'WARNING. EXTRAPOLATION OF SVP PERFORMED.')
      IEX=1
60    CONTINUE
C     *** SAVE SPEED IN MEDIUM 1
      C1=CI
C     *** IS RECEIVER ON OR WITHIN 1 DZ OF INTERFACE?
      IF(ZLYR(ILYR)-ZI.GE.DZ) GO TO 65
C     *** YES. IS THIS BOTTOM LAYER?
      IF(ILYR.EQ.NLYR) GO TO 65
C     *** NO. GET PARAMETERS FOR MEDIUM 2
      R2=RHO(ILYR+1)
      BETA2=BETA(ILYR+1)
      C2=CSVP(IXSVP(ILYR)+1)
      GO TO 70
65    CONTINUE
C     *** NOT AN INTERFACE. USE MEDIUM 1 PARAMETERS.
      C2=C1
      R2=R1
      BETA2=BETA1
C     *** COMPUTE DENSITY RATIO
70    R12(I)=R1/R2
C     *** THE NEXT FEW LINES COMPUTE THE X DIAGONAL
      XN=C0/C1
      IF(BETA1.LT.0.0) BETA1=ALPHA*C1/FRQ
      XN1=CMPLX(XN*XN,XN*XN*BETA1/27.287527)
      XN=C0/C2
      IF(BETA2.LT.0.0) BETA2=ALPHA*C2/FRQ
      XN2=CMPLX(XN*XN,XN*XN*BETA2/27.287527)
      B=(-XK0*.5*((XN1-1.)+R12(I)*(XN2-1.0)))
      X(I)=DZ*DZ*XK0*(B-CMPLX(0.0,+2.0*(1.0+R12(I))/DR))+1+R12(I)
100   CONTINUE
C
C     *** COMPUTE BTA(N1) THROUGH BTA(N)
C     *** ARRAY BTA CONTAINS PARTIAL SOLUTION OF SYSTEM OF EQUATIONS
      IF(N1.EQ.1) GO TO 105
      M=N1
      GO TO 106
```

```
105    BTA(1)=X(1)
       M=2
105    DO 110 I=M,N
       BTA(I)=X(I)-R12(I-1)/BTA(I-1)
110    CONTINUE
C
C      *** COMPUTE 'A' COEFFICIENTS AT BOTTOM
       ACOFY=ACOFX
       ACOFX=CMPLX(0.0,0.5)*XK0*(XN2-1.0)
       RETURN
       END
```

```
      SUBROUTINE CRNK
C     ****************************************************************
C     ****************************************************************
C     * THIS ROUTINE USES THE CRANK-NICOLSON METHOD FOR SOLVING THE    *
C     * SYSTEM OF EQUATIONS. THE SOLUTION IS ADVANCED FROM RANGE RA-DR *
C     * TO RANGE RA WHERE RA IS THE ADVANCED RANGE.                    *
C     ****************************************************************
C     ****************************************************************
C     *** SYSTEM OF EQUATIONS IS AS FOLLOWS:
C
C          ALL VALUES ARE AT ADVANCED RANGE
C     ***                         ***     *** ***     *** ***
C     * X(1)  -R12(1)     0       *       * U(1) *    * SURX *
C     *  -1    X(2)    -R12(2) *    .   * U(2) *  -  *  0   *    =
C     *   0     -1      X(N)     *       * U(N) *    * BOTX *
C     ***                         ***     *** ***     *** ***
C
C          ALL VALUES ARE AT PRESENT RANGE
C     ***                         ***     *** ***     *** ***
C     * Y(1)  +R12(1)     0       *       * U(1) *    * SURY *
C     *  +1    Y(2)    +R12(2) *    .   * U(2) *  +  *  0   *
C     *   0     +1      Y(N)     *       * U(N) *    * BOTY *
C     ***                         ***     *** ***     *** ***
C
C     ****************************************************************
C     ****************************************************************
C
C         D      - ARRAY - RIGHT HAND SIDE
C         RA     - RANGE TO WHICH SOLUTION IS TO BE ADVANCED - METERS
C                - RA IS INCREMENTED BY DR PRIOR TO ENTERING THIS ROUTINE
C         TA     - TEMPORARY VARIABLE USED IN MANIPULATION OF BOTTOM POINT
C         TB     - TEMPORARY VARIABLE USED IN MANIPULATION OF BOTTOM POINT
C
C     *** SEE MAIN PROGRAM FOR OTHER DEFINITIONS
C     *** UNDEFINED VARIABLES ARE TEMPORARY VARIABLES
C     *** SUBROUTINE CRNK RETURNS:
C         N      - NUMBER OF EQUI-SPACED POINTS IN U AT RANGE RA
C         U      - ARRAY - COMPLEX ACOUSTIC PRESSURE FIELD AT RANGE RA
C                - INCLUDES BOTTOM POINT - DOES NOT INCLUDE SURFACE POINT
C     *** SUBROUTINE TRID IS CALLED TO SOLVE THE TRIDIAGONAL MATRIX
C     ****************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
      DIMENSION D(MXN)
      COMPLEX D,SUM,A,EX,P,Q,XX,S1,S2,CA,CB,CC,CD,TA,TB
```

```
C
C     *** CALL SCON FOR SURFACE CONDITION
      CALL SCON
C
C     *** COMPUTE RIGHT HAND SIDE D(1) THROUGH D(N-1)
C     *** D(N) IS COMPUTED LATER
      D(1)=Y(1)*U(1)+R12(1)*U(2)+SURY+SURX
      DO 5 I=2,N-1
5     D(I)=U(I-1)+Y(I)*U(I)+R12(I)*U(I+1)
C
C     *** BOTTOM TYPE
      IB=ITYPEB+1
      GO TO (10,40,80,82) ,IB
C
C     *** BOTTOM IS RIGID - ITYPEB = 0
10    CONTINUE
      IF(THETA.GT.0.0) GO TO 30
      IF(THETA.NE.0.0) GO TO 15
C
C     *** RIGID BOTTOM IS FLAT
      BOTY=U(N)
      D(N)=U(N-1)+Y(N)*U(N)+BOTY
      TA=0.0
C     *** BOTX=U(N) AT ADVANCED RANGE
      TB=1.0
      CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
      RETURN
C
C     *** RIGID BOTTOM SLOPES UPWARD - DELETE A POINT - USE 2ND ORDER ODE
15    BOTY=U(N)
      N=N-1
      D(N)=U(N-1)+Y(N)*U(N)+BOTY
      COT=COS(THETA)/SIN(THETA)
      P=-COT/BCOF
      Q=(ACOFX+CMPLX(0.0,XKO))/BCOF
      XX=CSQRT(P*P-4.0*Q)
      S1=(-P+XX)/2.0
      S2=(-P-XX)/2.0
      CA=(1.0-S1*DZ)/(-XX*DZ)
      CB=1.0/(XX*DZ)
      CC=1.0-CA
      CD=CB
      TA=-CD*CEXP(S1*DZ)+CB*CEXP(S2*DZ)
      TB=CC*CEXP(S1*DZ)+CA*CEXP(S2*DZ)
      CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
      RETURN
C
30    CONTINUE
C     *** RIGID BOTTOM SLOPES DOWNWARD - USE 2ND ORDER O.D.E.
      COT=COS(THETA)/SIN(THETA)
      P=-COT/BCOF
      Q=(ACOFY+CMPLX(0.0,XKO))/BCOF
      XX=CSQRT(P*P-4.0*Q)
      S1=(-P+XX)/2.0
```

```
      S2=(-P-XX)/2.0
      CA=(1.0-S1*DZ)/(-XX*DZ)
      CB=1.0/(XX*DZ)
      CC=1.0-CA
      CD=CB
      TA=-CD*CEXP(S1*DZ)+CB*CEXP(S2*DZ)
      TB=CC*CEXP(S1*DZ)+CA*CEXP(S2*DZ)
C
C     *** MODIFY LOWER AND Y DIAGONALS IN ROW N BY TA AND TB  - THEN
C     *** COMPUTE D(N)
      D(N)=(1.0+TA)*U(N-1)+(Y(N)+TB)*U(N)
C
C     *** COMPUTE TA AND TB FOR LOWER AND X DIAGONALS IN ROW N
      Q=(ACOFX+CMPLX(0.0,XK0))/BCOF
      XX=CSQRT(P*P-4.0*Q)
      S1=(-P+XX)/2.0
      S2=(-P-XX)/2.0
      CA=(1.0-S1*DZ)/(-XX*DZ)
      CB=1.0/(XX*DZ)
      CC=1.0-CA
      CD=CB
      TA=-CD*CEXP(S1*DZ)+CB*CEXP(S2*DZ)
      TB=CC*CEXP(S1*DZ)+CA*CEXP(S2*DZ)
      CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
C
C     *** ADD A POINT
      CB=((U(N)-U(N-1))/DZ-S1*U(N))/(-XX)
      CA=U(N)-CB
      N=N+1
      U(N)=CA*CEXP(S1*DZ)+CB*CEXP(S2*DZ)
      RETURN
C
40    CONTINUE
C
C     *** BOTTOM CONDITION SUPPLIED BY USER - ITYPEB = 1
      IF(THETA.GT.0.0) GO TO 60
      IF(THETA.NE.0.0) GO TO 70
C
C     *** FLAT BOTTOM
C     *** USER SUPPLIES BOTY ON BOTTOM INTERFACE AT PRESENT RANGE
C     *** USER SUPPLIES BOTX ON BOTTOM INTERFACE AT ADVANCED RANGE
50    CALL BCON
      TA=0.0
      TB=0.0
      N=N-1
      D(N)=U(N-1)+Y(N)*U(N)+BOTY+BOTX
      CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
      N=N+1
      U(N)=BOTX
      RETURN
C
60    CONTINUE
C     *** LAYER SLOPES DOWN
C     *** USER SUPPLIES :
```

```
C           BOTY - ONE DZ IN BOTTOM AT PRESENT RANGE
C           BOTX - ON BOTTOM INTERFACE AT ADVANCED RANGE
        CALL BCON
        TA=0.0
        TB=0.0
        D(N)=U(N-1)+Y(N)*U(N)+BOTY+BOTX
        CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
        N=N+1
        U(N)=BOTX
        RETURN
C
70      CONTINUE
C       *** LAYER SLOPES UP
C       *** USER SUPPLIES :
C           BOTX - ONE DZ BELOW BOTTOM INTERFACE AT ADVANCED RANGE
        CALL BCON
        N=N-1
        D(N)=U(N-1)+Y(N)*U(N)+BOTY+BOTX
        TA=0.0
        TB=0.0
        CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
        RETURN
C
C
C       *** ITYPEB = 2 -- ARTIFICIAL ABSORBING LAYER INTRODUCED
80      CONTINUE
        IF(THETA.NE.0.0) GO TO 85
C
C       *** BOTTOM OF ABSORBING LAYER IS FLAT
C       *** ITYPEB = 2 OR 3 -- ARTIFICIAL ABSORBING LAYER
82      U(N)=0.0
        TA=0.0
        TB=0.0
        CALL TRID(X,U,D,BTA,R12,N-1,N,TA,TB)
        RETURN
C
C       *** ITYPEB = 2  -- ARTIFICIAL ABSORBING LAYER NOT FLAT
85      IF(THETA.GT.0.0) GO TO 90
C
C       *** BOTTOM OF ABSORBING LAYER  SLOPES UP
        TA=0.0
        TB=0.0
        U(N-1)=0.0
        U(N)=0.0
        CALL TRID(X,U,D,BTA,R12,N-2,N-1,TA,TB)
C
C       *** DELETE A POINT
        N=N-1
        RETURN
C
90      CONTINUE
C       *** BOTTOM OF ABSORBING LAYER SLOPES DOWN
        D(N)=U(N-1)
        TA=0.0
```

```
      TB=0.0
      CALL TRID(X,U,D,BTA,R12,N,N,TA,TB)
C     *** ADD A POINT
      N=N+1
      U(N)=0.0
      RETURN
      END
```

```
      SUBROUTINE TRID(X,U,D,BTA,R12,L,M,TA,TB)
C     ******************************************************************
C     *** SPECIALIZED VERSION OF METHOD PRESENTED ON PG 442 OF CARNAHAN
C     *** ET AL FOR SOLVING A TRIDIAGONAL MATRIX.
C     *** TRID RETURNS THE SOLUTION FIELD IN U
C     *** LOWER DIAGONAL = -1
C     *** UPPER DIAGONAL = -R12
C     *** BTA IS PARTIAL SOLUTION COMPUTED IN ROUTINE DIAG
C     *** D CONTAINS R.H.S.
C     *** GAMMA - TEMPORARY STORAGE
C     *** TA    - TEMPORARY VARIABLE USED IN MANIPULATION OF BOTTOM POINT
C     *** TB    - TEMPORARY VARIABLE USED IN MANIPULATION OF BOTTOM POINT
C     ******************************************************************
C
      PARAMETER MXN=10000
      DIMENSION X(1),U(1),GAMMA(MXN),D(1),BTA(1),R12(1)
      COMPLEX X,U,GAMMA,D,BTA,TA,TB
C     *** X AND LOWER DIAGONAL IN ROW L MUST BE MODIFIED BY TB AND TA
      BTA(L)=(X(L)-TB)-((1.0+TA)*R12(L-1))/BTA(L-1)
      GAMMA(1)=D(1)/BTA(1)
      DO 10 I=2,L
      GAMMA(I)=(D(I)+GAMMA(I-1))/BTA(I)
10    CONTINUE
      GAMMA(L)=GAMMA(L)+(TA*GAMMA(L-1))/BTA(L)
C     *** IF L IS LESS THAN M, U(M) IS KNOWN.
      IF(L.EQ.M) U(M)=GAMMA(M)
      DO 70 I=M,2,-1
      U(I-1)=GAMMA(I-1)+R12(I-1)*U(I)/BTA(I-1)
70    CONTINUE
      RETURN
      END
```

```
      COMPLEX FUNCTION HNKL(X)
C     ****************************************************************
C     * HANKEL FUNCTION HO(1) - POLYNOMIAL APPROXIMATION             *
C     * HANDBOOK OF MATH FUNCTIONS - N.B.S. - NOV 1967               *
C     ****************************************************************
C
      REAL JO
      DATA PI/3.141592654/
C
      IF(X.GT.3.) GO TO 10
C
C     *** (-3.0.LE.X.LE.3.0)
      Y=X*X/9.0
      JO=1.+Y*(-2.2499997+Y*(+1.2656208+Y*(-0.3163866+Y*(+0.0444479
     C     +Y*(-0.0039444+Y*(+0.0002100))))))
C
C     *** (0.0.LT.X.LE.3.0)
      YO=2.0*LOG(0.5*X)*JO/PI+0.36746691
     C     +Y*(+0.60559366+Y*(-0.74350384+Y*(+0.25300117+Y*(-0.04261214
     C     +Y*(+0.00427916+Y*(-0.00024846))))))
      HNKL=CMPLX(JO,YO)
      RETURN
C
C     *** (3.0.LE.X.LT.INFINITY)
10    Y=3.0/X
      FO=  0.79788456+Y*(-0.00000077+Y*(-0.00552740+Y*(-0.00009512
     C                +Y*(+0.00137237+Y*(-0.00072805+Y*(+0.00014476))))))
      TO=X-0.78539816+Y*(-0.04166397+Y*(-0.00003954+Y*(+0.00262573
     C                +Y*(-0.00054125+Y*(-0.00029333+Y*(+0.00013558))))))
      HNKL=FO*CEXP(CMPLX(0.0,TO))/SQRT(X)
      RETURN
      END
```

```
      SUBROUTINE SVP(NLYR,ZLYR,RHO,BETA,IXSVP,NSVP,ZSVP,CSVP)
C     ****************************************************************
C     *** SOUND VELOCITY PROFILE SUBROUTINE
C     *** CALLING PROGRAM SUPPLIES: NOTHING
C     *** SVP SUBROUTINE RETURNS:
C         NLYR - NUMBER OF LAYERS. L YER 1 IS WATER. OTHERS ARE SEDIMENT
C         ZLYR - ARRAY - DEPTH OF EACH LAYER. FIRST IS DEPTH OF WATER.
C         RHO  - ARRAY - DENSITY OF EACH LAYER. GRAMS/CUBIC CM
C         BETA - ARRAY - ATTENUATION IN EACH LAYER. DB/WAVELENGTH
C         IXSVP- ARRAY - CONTAINS POINTERS. POINTS TO LAST VALUE OF SVP
C                IN CORRESPONDING LAYER. SVP IS STORED IN ARRAYS ZSVP
C                AND CSVP. IXSVP(1) POINTS TO LAST SVP POINT IN WATER.
C                IXSVP(NLYR) POINTS TO LAST SVP POINT IN BOTTOM-MOST LAYER.
C         NSVP - NUMBER OF POINTS IN ZSVP AND CSVP. ZSVP AND CSVP
C                CONTAIN THE PROFILES FOR ALL LAYERS.
C         ZSVP - ARRAY - SVP DEPTHS - METERS
C         CSVP - ARRAY - SOUND SPEED - METERS/SEC
C     ****************************************************************
C
      PARAMETER NIU=1,NPU=6
      DIMENSION ZLYR(1),RHO(1),BETA(1),IXSVP(1),ZSVP(1),CSVP(1)
C
      NSVP=0
C     *** READ NUMBER OF LAYERS
      READ(NIU,*,END=100) NLYR
C     *** FIRST LAYER IS WATER. OTHERS ARE SEDIMENT.
      DO 55 I=1,NLYR
C     *** READ DEPTH OF LAYER, DENSITY AND ATTENUATION.
      READ(NIU,*,END=100) ZLYR(I),RHO(I),BETA(I)
C     *** READ PROFILE.
50    READ(NIU,*,END=100) ZV,CV
      NSVP=NSVP+1
      ZSVP(NSVP)=ZV
      CSVP(NSVP)=CV
      IF(ZV.LT.ZLYR(I)) GO TO 50
      IF(ZV.GT.ZLYR(I)) GO TO 100
      IXSVP(I)=NSVP
55    CONTINUE
      RETURN
C
C     *** ERROR EXIT
100   NSVP=0
      RETURN
      END
```

```
      SUBROUTINE SFIELD(FRQ,CO,ZS,N,DZ,U)
C     *****************************************************************
C     *** GAUSSIAN STARTING FIELD - SEE NORDA TECH NOTE 12 BY H.K.BROCK
C     *** SFIELD IS CALLED IF INPUT PARAMETER ISF = 0
C     *****************************************************************
C
C     *** CALLING ROUTINE SUPPLIES:
C         FRQ - FREQUENCY IN HZ
C         CO  - REFERENCE SOUND SPEED - METERS/SEC
C         ZS  - DEPTH OF SOURCE IN METERS.
C         N   - NUMBER OF POINTS IN ARRAY U
C         DZ  - DEPTH INCREMENT - METERS
C     *** SFIELD SUBROUTINE SUPPLIES:
C         U   - COMPLEX STARTING FIELD
C     *****************************************************************
C
      COMPLEX U(1)
      DATA PI/3.1415926535/
C
C     THE FIELD IS DEFINED AS A GAUSSIAN BEAM AT RANGE = 0.
C     LOCAL VARIABLES - GA  GAUSSIAN AMPLITUDE
      XKO=2.0*PI*FRQ/CO
      GW=2.0/XKO
      GA=SQRT(GW)/GW
      DO 10 I=1,N
      ZM=I*DZ
      PR=GAUSS(GA,ZM,ZS,GW)-GAUSS(GA,-ZM,ZS,GW)
      U(I)=CMPLX(PR,0.0)
10    CONTINUE
      RETURN
      END
      FUNCTION GAUSS(GA,Z,GD,GW)
C     INPUT  - GA  GAUSSIAN AMPLITUDE
C     OUTPUT - GAUSS = GA * EXP(-((Z - GD) / GW)**2)
C     TEMPORARY VARIABLE - V
      V=(Z-GD)/GW
      V=-(V*V)
      GAUSS=GA*EXP(V)
      RETURN
      END
```

```
      SUBROUTINE USVP
C     ****************************************************************
C     *** USER SOUND VELOCITY PROFILE SUBROUTINE
C         SUBROUTINE USVP IS CALLED EACH DR IN RANGE AS LONG AS
C         KSVP IS NOT ZERO. KSVP MAY BE USED BY USER TO TRANSFER CONTROL
C         IN THIS SUBROUTINE. USER INSERTS LOGIC TO CLEAR KSVP
C         WHEN USVP IS NO LONGER NEEDED. IF KSVP NOT CLEARED BY USER,
C         USVP IS CALLED EACH STEP IN RANGE UNTIL RA = NEXT RSVP.
C     ****************************************************************
C     *** USVP SUBROUTINE RETURNS:
C         NLYR - NUMBER OF LAYERS. LAYER 1 IS WATER. OTHERS ARE SEDIMENT
C         ZLYR - ARRAY - DEPTH OF EACH LAYER. FIRST IS DEPTH OF WATER.
C         RHO  - ARRAY - DENSITY OF EACH LAYER. GRAMS/CUBIC CM
C         BETA - ARRAY - ATTENUATION IN EACH LAYER. DB/WAVELENGTH
C         IXSVP - ARRAY - CONTAINS POINTERS. POINTS TO LAST VALUE OF SVP
C                IN CORRESPONDING LAYER. SVP IS STORED IN ARRAYS ZSVP
C                AND CSVP. IXSVP(1) POINTS TO LAST SVP POINT IN WATER.
C         NSVP - NUMBER OF POINTS IN ZSVP AND CSVP. ZSVP AND CSVP
C                CONTAIN THE PROFILES FOR ALL LAYERS.
C         ZSVP - ARRAY - SVP DEPTHS - METERS
C         CSVP - ARRAY - SOUND SPEED - METERS/SEC
C         KSVP - AS DESCRIBED ABOVE.
C     ****************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      GO TO (100,200,300,400) ,KSVP
      NSVP=0
      RETURN
C
100   CONTINUE
C
C     *** IF KSVP=1, CONTROL IS TRANSFERRED HERE. USER LOADS
C         NLYR,ZLYR(I),RHO(I),BETA(I), AND IXSVP(I) WHERE I=1,NLYR.
C         USER ALSO LOADS NSVP,ZSVP(I), AND CSVP(I) WHERE I=1,NSVP.
C         KSVP MAY BE ALTERED DEPENDING ON USER LOGIC.
C
C     *** USER SUPPLIES SVP
C     *** SETUP FOR ONE LAYER WITH FOUR SVP POINTS SHOWN BELOW
C
      NLYR=1
C     ZLYR(1)=.....
      RHO(1)=1.0
C     BETA(1)=.....
      NSVP=4
```

```
C       ZSVP(1)=.....
C       CSVP(1)=.....
C       ZSVP(2)=.....
C       CSVP(2)=.....
C       ZSVP(3)=.....
C       CSVP(3)=.....
C       ZSVP(4)=.....
C       CSVP(4)=.....
        IXSVP(1)=NSVP
        RETURN
C
200     CONTINUE
C       *** USER INSERTS CODE HERE IF DESIRED
        RETURN
C
300     CONTINUE
C       *** USER INSERTS CODE HERE IF DESIRED
        RETURN
C
400     CONTINUE
C       *** USER INSERTS CODE HERE IF DESIRED
        RETURN
        END
```

```
      SUBROUTINE UFIELD
C     ****************************************************************
C     *** USER STARTING FIELD
C     *** USER WRITES THIS SUBROUTINE IF GAUSSIAN FIELD NOT DESIRED
C     *** UFIELD IS CALLED IF INPUT PARAMETER ISF IS NOT ZERO
C     ****************************************************************
C     *** UFIELD SUBROUTINE SUPPLIES:
C         U   - COMPLEX STARTING FIELD
C     ****************************************************************
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
C     *** STARTING FIELD GENERATED BY USER
      DO 10 I=1,N
      ZI=I*DZ
C     U(I)=.....
10    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BCON
C     ****************************************************************
C     *** USER PREPARED BOTTOM CONDITION SUBROUTINE
C         BCON IS CALLED IF INPUT PARAMETER ITYPEB = 1
C         SEE MAIN PROGRAM FOR DEFINITIONS
C     ****************************************************************
C     *** SUBROUTINE RETURNS:
C         BOTY,BOTX
C     ****************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      IF(THETA) 50,100,150
C
C     *** THETA LESS THAN 0.0. BOTTOM SLOPES UP.
50    CONTINUE
      BOTY=U(N)
C     BOTX=..............
      RETURN
C
C     *** THETA = 0.0. BOTTOM IS FLAT.
100   CONTINUE
      BOTY=U(N)
C     BOTX=...............
      RETURN
C
C     *** THETA GREATER THAN 0.0, BOTTOM SLOPES DOWN.
150   CONTINUE
C     BOTY=..............
C     BOTX=..............
      RETURN
      END
```

```
      SUBROUTINE SCON
C     ***************************************************************
C     *** SURFACE CONDITION SUBROUTINE
C         IF ITYPES = 0, SCON SETS SURY AND SURX = 0.0.
C         IF ITYPES NOT 0, THE USER MUST SUPPLY SURY AND SURX.
C         SEE MAIN PROGRAM FOR DEFINITIONS
C     ***************************************************************
C
      PARAMETER MXLYR=101,MXN=10000,MXSVP=101,MXTRK=101,NIU=1,
     C          NOU=2,NPU=6
      COMPLEX ACOFX,ACOFY,BCOF,BOTX,BOTY,BTA,HNK,HNKL,SURX,SURY,TEMP,
     C        U,X,Y
      COMMON /IFDCOM/ACOFX,ACOFY,ALPHA,BCOF,BETA(MXLYR),BOTX,BOTY,
     C       BTA(MXN),CO,CSVP(MXSVP),DR,DR1,DZ,FRQ,IHNK,ISF,ITYPEB,
     C       ITYPES,IXSVP(MXLYR),KSVP,N,N1,NLYR,NSVP,NWSVP,R12(MXN),RA,
     C       RHO(MXLYR),RSVP,SURX,SURY,THETA,TRACK(MXTRK,2),U(MXN),
     C       X(MXN),XKO,Y(MXN),ZA,ZLYR(MXLYR),ZP,ZS,ZSVP(MXSVP)
      DATA PI/3.141592654/,DEG/57.29578/
C
      IF(ITYPES.NE.0) GO TO 100
C
C     *** PRESSURE RELEASE SURFACE
      SURY=0.0
      SURX=0.0
      RETURN
C
C     *** USER SURFACE CONDITION
100   CONTINUE
C     SURY=..........
C     SURX=..........
      RETURN
      END
```

Appendix B

PLOT PROGRAM COMPUTER LISTING

```
C     *******************************************************************
C     *******************************************************************
C     *** PLOT PROGRAM FOR IFD MODEL.
C     *** PROGRAM PLOTS PROPAGATION LOSS VS RANGE ON CALCOMP PLOTTER -
C     *** MODEL 1039.
C     *** SEE MAIN PROGRAM IFD FOR DEFINITIONS OF IFD VARIABLES.
C     *******************************************************************
C     *******************************************************************
C     *** INPUT
C     *******************************************************************
C         INPUT UNIT NUMBER = NIU
C         INPUT FILE NAME   = PLTIFD.IN
C         CONTENTS:  CARD IMAGES IN FREE FORMAT
C         CARD 1  :  Z1,Z2,Z3,Y1,Y2,Y3,YL,X1,X2,X3,XL,FACT,XAVG
C           .     :  .         .         .         .         .
C         CARD N  :  .         .         .         .         .
C     *******************************************************************
C     *** QUICK REFERENCE AND NOTES FOR CARD INPUT
C     *******************************************************************
C         Z1 = FIRST RECEIVER DEPTH TO PLOT - METERS
C              IF (Z1.LE.0.0) PROGRAM TERMINATES
C         Z2 = LAST RECEIVER DEPTH TO PLOT  - METERS
C         Z3 = RECEIVER DEPTH INCREMENT - METERS
C         Y1 = LABEL OF Y-AXIS AT ORIGIN IN DB
C         Y2 = LABEL AT TOP OF Y-AXIS IN DB
C         Y3 = INCREMENT OF Y-AXIS LABELS IN DB
C         YL = LENGTH OF Y-AXIS IN INCHES
C         X1 = LABEL OF X-AXIS AT ORIGIN IN KILOMETERS
C         X2 = LABEL AT RIGHT OF X-AXIS IN KILOMETERS
C         X3 = INCREMENT OF X-AXIS LABELS IN KILOMETERS
C         XL = LENGTH OF X-AXIS IN INCHES
C         FACT = SCALE OF PLOT: 1.0 = FULL SIZE ; .5 = 1/2 SIZE ; ETC.
C         XAVG = RANGE OVER WHICH TO COMPUTE RUNNING AVERAGE IN METERS
C                IF XAVG = 0, ALL POINTS ARE PLOTTED
C     *******************************************************************
C     *******************************************************************
      PARAMETER MAXP=5000
      PARAMETER MXLYR=101,MXN=10000,NIU=1,NOU=2,NPU=6,PLTU=3
      COMPLEX HNK,HNKL,CTEMP,U(MXN)
      DIMENSION BETA(MXLYR),RHO(MXLYR),ZLYR(MXLYR),IBUF(2000)
      DIMENSION P(MAXP),R(MAXP)
      DATA PI/3.141592654/,DEG/57.29578/
      DATA CNVKM/1000.0/
      IPRNT=0
C     *** ASSIGN IFD OUTPUT FILE
      CALL ASSIGN(NOU,'IFD.OUT')
C     *** ASSIGN PLOT PARAMETER INPUT FILE
      CALL ASSIGN(NIU,'PLTIFD.IN')
      CALL PLOTS(IBUF,2000,PLTU)
      CALL PLOT (0.0,0.5,-3)
C     *** READ PLOT PARAMETERS
100   READ(NIU,*,END=510) Z1,Z2,Z3,Y1,Y2,Y3,YL,X1,X2,X3,XL,FACT,XAVG
      IF(Z1.LE.0.0) GO TO 510
      IF(FACT.LE.0.0) FACT=1.0
      CALL FACTOR(FACT)
```

```
      YINC=(Y2-Y1)/YL
      IF(Z3.EQ.0.0) Z3=1.0
C     *** GENERATE PLOT FOR RECEIVER DEPTH
      DO 350 ZR=Z1,Z2,Z3
      ZRR=ZR
      IPEN=3
      IX=1
      DX=(X2-X1)/XL
      CALL AXIS2(0.,0.,'RANGE (KM)',-10,XL,0.,X1,X3,X2)
      CALL AXIS2(XL,0.,' ',-1,YL,90.,Y1,Y3,Y2)
      CALL AXIS2(0.,0.,'PROPLOSS (DB)',+13.YL,90.,Y1,Y3,Y2)
      CALL PLOT(0.0,YL,3)
      CALL PLOT(XL,YL,2)
C     *** READ INITIAL IFD PARAMETERS
      REWIND(NOU)
      READ(NOU) FRQ,ZS,C0,ISF,R0,Z0,N,IHNK,ITYPEB,ITYPES,RMAX,DR,WDR,DZ,
     CNLYR,ZLYR,RHO,BETA
      IF(XAVG.LT.WDR) XAVG=WDR
      L=0
115   CONTINUE
      RAVG=0.0
      PLAVG=0.0
C     *** READ SOLUTION FIELD
120   READ(NOU,END=170)NN,RA,WDZ,(U(I),I=1,NN)
      IF(RA.LE.0.0) GO TO 120
      HNK=HNKL(2.0*PI*FRQ*RA/C0)
      I=0
      INTERP=0
      I=ZRR/WDZ
      IF(I.GT.NN) GO TO 115
      IF(I.GE.1) GO TO 130
      I=1
      ZRR=WDZ
130   IF(I*WDZ.NE.ZRR) INTERP=1
      Y=CABS(U(I))
      IF(IHNK.NE.0) Y=CABS(U(I)*HNK)
      IF(INTERP.EQ.1) CTEMP=U(I)+(U(I+1)-U(I))*(ZRR-I*WDZ)/WDZ
      IF(INTERP.EQ.1.AND.IHNK.NE.0) Y=CABS(CTEMP*HNK)
      IF(Y.LE.0.0) GO TO 120
      L=L+1
      P(L)=Y
      R(L)=RA
      GO TO 120
170   CONTINUE
      K=0
200   K=K+1
      RAVG=0
      PLAVG=0
      NAVG=0
      DO 210 J=K,L
      IF(R(J)-R(K).GE.XAVG) GO TO 220
      RAVG=RAVG+R(J)
      PLAVG=PLAVG+P(J)
      NAVG=NAVG+1
210   CONTINUE
220   CONTINUE
      IF(NAVG.EQ.0) GO TO 250
      BIAS=0.0
```

```
      RA=RAVG/NAVG
      IF(IHNK.EQ.0.AND.RA.GT.0.0) BIAS=10.0*ALOG10(RA)
      Y=PLAVG/NAVG
      IF(Y.LE.0.0) GO TO 200
      Y=-20.0*ALOG10(Y)+BIAS
      TEMP=RA/1000.0
      IF(IPRNT.EQ.1) WRITE(NPU,*) TEMP,Y
      IF(RA/CNVKM.GT.X2-XAVG/CNVKM) GO TO 250
      IF(RA/CNVKM.LT.X1) GO TO 200
      X=(RA/CNVKM-X1)/DX
      Y=(Y-Y1)/YINC
      IF(Y.LT.0.0) Y=0.0
      IF(Y.GT.YL) Y=YL
      CALL PLOT(X,Y,IPEN)
      IPEN=2
      GO TO 200
250   CONTINUE
      CALL BLOCK(XL,YL,FRQ,ZS,C0,ISF,R0,Z0,N,IHNK,ITYPEB,ITYPES,
     CRMAX,DR,WDR,WDZ,DZ,ZRR,NLYR,ZLYR,BETA,RHO,XAVG)
      CALL PLOT(XL+2.0,0.0,-3)
350   CONTINUE
      GO TO 100
510   CONTINUE
      CALL PLOT(0.0,-0.5,999)
      STOP
      END

      SUBROUTINE BLOCK(XL,YL,FRQ,ZS,C0,ISF,R0,Z0,N,IHNK,ITYPEB,ITYPES,
     CRMAX,DR,WDR,WDZ,DZ,ZRR,NLYR,ZLYR,BETA,RHO,XAVG)
      DIMENSION ZLYR(1),BETA(1),RHO(1)
      NC=30 ! MAX CHAR IN STRING
      HT=.1
      DY=1.5*HT
      XBLK=5.0*HT
      YBLK=YL+(21+NLYR)*DY
      CALL SYMBOL(XBLK,YBLK,HT,'IFD SOLUTION',0.,12)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'INITIAL PARAMETERS',0.0,18)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'FRQ = ',0.,6)
      CALL NUMBER(999.,YBLK,HT,FRQ,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' HZ',0.,3)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'ZS = ',0.,5)
      CALL NUMBER(999.,YBLK,HT,ZS,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'C0 = ',0.,5)
      CALL NUMBER(999.,YBLK,HT,C0,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M/SEC',0.,6)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'R0 = ',0.,5)
      CALL NUMBER(999.,YBLK,HT,R0,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'Z0 = ',0.,5)
      CALL NUMBER(999.,YBLK,HT,Z0,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
```

```
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'N =  ',0.,5)
      FN=N
      CALL NUMBER(999.,YBLK,HT,FN,0.,-1)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'DR = ',0.,5)
      CALL NUMBER(999.,YBLK,HT,DR,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'WDR = ',0.,6)
      CALL NUMBER(999.,YBLK,HT,WDR,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'RMAX = ',0.,7)
      CALL NUMBER(999.,YBLK,HT,RMAX,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'DZ = ',0.,5)
      CALL NUMBER(999.,YBLK,HT,DZ,0.,2)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'WDZ = ',0.,6)
      CALL NUMBER(999.,YBLK,HT,WDZ,0.,2)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'ISF = ',0.,6)
      FPN=ISF
      CALL NUMBER(999.,YBLK,HT,FPN,0.,-1)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'IHNK = ',0.,7)
      FPN=IHNK
      CALL NUMBER(999.,YBLK,HT,FPN,0.,-1)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'ITYPES = ',0.,9)
      FPN=ITYPES
      CALL NUMBER(999.,YBLK,HT,FPN,0.,-1)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'ITYPEB = ',0.,9)
      FPN=ITYPEB
      CALL NUMBER(999.,YBLK,HT,FPN,0.,-1)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'LYR  DEPTH(M) RHO    BETA(DB/WL)',0.,32)
      DO 500 I=1,NLYR
      YBLK=YBLK-DY
      FPN=I
      CALL NUMBER(XBLK,YBLK,HT,FPN,0.,-1)
      CALL NUMBER(XBLK+5.0*HT,YBLK,HT,ZLYR(I),0.,1)
      CALL NUMBER(XBLK+14.0*HT,YBLK,HT,RHO(I),0.,2)
      IF(BETA(I).GE.0.0)CALL NUMBER(XBLK+21.0*HT,YBLK,HT,BETA(I),0.,3)
      IF(BETA(I).LT.0.)CALL SYMBOL(XBLK+21.0*HT,YBLK,HT,'COMPUTED',0.,8)
  500 CONTINUE
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'AVG =  ',0.,7)
      CALL NUMBER(999.,YBLK,HT,XAVG,0.,-1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      YBLK=YBLK-DY
      CALL SYMBOL(XBLK,YBLK,HT,'RECEIVER DEPTH = ',0.,17)
      CALL NUMBER(999.,YBLK,HT,ZRR,0.,1)
      CALL SYMBOL(999.,YBLK,HT,' M',0.,2)
      RETURN
      END
```

INITIAL DISTRIBUTION LIST

Addressee	No. of Copies
ONR (Codes 100, 200, 480, 222, 486)	5
CNO (OP-952)	1
CNM (MAT-08T24)	1
NAVELECSYSCOM (PME-124, ELEX 304)	2
NAVSEASYSCOM (SEA-003, -06R, -63R, -63R-1, -63R-13, -63RA)	6
DTIC	12
Dr. Ralph N. Baer, Code 5160, NRL, Washington, D.C. 20375	1
Dr. Homer Bucker, Code 5311B, NOSC, San Diego, CA 92152	1
Prof. Robert A. Koch, Applied Research Laboratories, University of Texas at Austin, Austin, Texas 78712	1
Dr. W.A. Kuperman, NORDA, NSTL Station, MS 39529	1
Dr. S. McDaniel, Applied Research Lab., The Pennsylvania State University, University Park, PA 16802	1
Dr. Lewis Dozier, Science Applications, Inc., 1710 Goodridge Drive, McLean, VA 22180	1
Paul Etter, MAR Inc., 1335 Rockville Pike, Rockville, MD 20852	1
David Gordon, NOSC, San Diego, CA 92152	1
Dr. Robert Greene, Science Applications, Inc., 1710 Goodridge Dr., P.O. Box 1303, McLean, VA 22101	1
Dr. David Palmer, Code 5122, National Oceanic & Atmospheric Adm., AOML, 15 Rickenbacker Causeway, Miami, FL 33149	1
M.A. Pedersen, Code 724, NOSC, San Diego, CA 92152	1
Mr. John S. Perkins, Code 5160, NRL, Washington, D.C. 20375	1
Prof. Morris Schulkin, 9325 Orchard Brook Dr., Potomac, MD 20854	1
Dr. F.B. Jensen, SACLANT ASW Research Center, 19026 LaSpezia, ITALY	1
Prof. Allan D. Pierce, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332	1
Prof. Martin H. Schultz, Dept. of Computer Science, Yale University, 10 Hillhouse Ave., New Haven, CT 06520	1
LCDR M.A. McAllister, NORDA, Code 520, NSTL Station, MS 39529	1
Dr. C.W. Spofford, SAI, 1710 Goodridge Dr., McLean, VA 22101	1
Prof. Fred Tappert, University of Miami, School of Marine & Atmospheric Science, 4600 Rickenbacker Causeway, Miami, FL 33149	1
Dr. John Tsai, AOML/Ocean Acoustics Lab., 4301 Rickenbacker Causeway, Miami, FL 33149	1
Prof. Jacob Yaniv, TEL AVIV Univ. (School of Engineering) RAMAT-AVIV, ISRAEL	1
R. Martin, NORDA, NSTL Station, MS 39529	1